

# LilyPond

---

Il compositore tipografico per la musica

## Guida alla Notazione

Il team di sviluppo di LilyPond

Questo manuale costituisce la guida di riferimento per tutti gli aspetti relativi alla notazione musicale in LilyPond versione 2.19.36. Si presuppone che il lettore conosca il materiale esposto nel Sezione “Manuale di Apprendimento” in *Manuale di Apprendimento*.

Per maggiori informazioni su come questo manuale si integra col resto della documentazione, o per leggere questo manuale in altri formati, si veda Sezione “Manuali” in *Informazioni generali*. Se ti manca qualche manuale, puoi trovare la completa documentazione all’indirizzo <http://www.lilypond.org/>.

Copyright © 1999–2015 degli autori.

*La traduzione della seguente nota di copyright è gentilmente offerta per le persone che non parlano inglese, ma solo la nota in inglese ha valore legale.*

*The translation of the following copyright notice is provided for courtesy to non-English speakers, but only the notice in English legally counts.*

E’ garantito il permesso di copiare, distribuire e/o modificare questo documento seguendo i termini della Licenza per Documentazione Libera GNU, Versione 1.1 o ogni versione successiva pubblicata dalla Free Software Foundation; senza alcuna sezione non modificabile. Una copia della licenza è acclusa nella sezione intitolata ”Licenza per Documentazione Libera GNU”.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections. A copy of the license is included in the section entitled “GNU Free Documentation License”.

Per la versione di LilyPond 2.19.36

---

# Sommario

<b>1</b>	<b>Notazione musicale</b>	<b>1</b>
1.1	Altezze	1
1.1.1	Inserimento delle altezze	1
	Ottava assoluta	1
	Ottava relativa	2
	Alterazioni	5
	Nomi delle note in altre lingue	8
1.1.2	Modifica di più altezze	9
	Controlli di ottava	10
	Trasposizione	11
	Inversione	14
	Retrogradazione	14
	Trasposizioni modali	15
1.1.3	Aspetto delle altezze	17
	Chiave	17
	Armatatura di chiave	22
	Segni di ottavazione	24
	Trasporto strumentale	26
	Alterazioni automatiche	28
	Ambitus	35
1.1.4	Teste di nota	37
	Teste di nota speciali	38
	Testa di nota con nome della nota	39
	Teste di nota a forma variabile	41
	Improvvisazione	44
1.2	Ritmi	44
1.2.1	Inserimento delle durate	45
	Durata	45
	Gruppi irregolari	48
	Scalare le durate	52
	Legature di valore	53
1.2.2	Inserimento delle pause	57
	Pause	57
	Pause invisibili	60
	Pause d'intero	61
1.2.3	Aspetto dei ritmi	65
	Indicazione di tempo	65
	Indicazioni metronomiche	70
	Anacrusi	73
	Musica in tempo libero	75
	Notazione polimetrica	76
	Divisione automatica delle note	79
	Mostrare i ritmi della melodia	81
1.2.4	Travature	83
	Travature automatiche	83
	Impostare il comportamento delle travature automatiche	86
	Travature manuali	95
	Travature a raggiera	98

1.2.5	Battute.....	99
	Stanghette.....	99
	Numeri di battuta.....	106
	Controlli di battuta e del numero di battuta.....	111
	Segni di chiamata.....	112
1.2.6	Questioni ritmiche particolari.....	114
	Abbellimenti.....	114
	Allineamento sulle cadenze.....	119
	Gestione del tempo.....	120
1.3	Segni di espressione.....	121
1.3.1	Segni di espressione collegati alle note.....	121
	Articolazioni e abbellimenti.....	121
	Dinamiche.....	124
	Nuove indicazioni dinamiche.....	130
1.3.2	Indicazioni espressive curvilinee.....	132
	Legature di portamento.....	133
	Legature di frase.....	136
	Respiri.....	137
	Portamenti indeterminati discendenti (cadute) e ascendenti.....	139
1.3.3	Indicazioni espressive lineari.....	140
	Glissando.....	140
	Arpeggio.....	144
	Trilli.....	148
1.4	Ripetizioni.....	150
1.4.1	Ripetizioni lunghe.....	150
	Ripetizioni normali.....	151
	Indicazioni di ripetizione manuali.....	158
	Ripetizioni ricopiate.....	160
1.4.2	Ripetizioni brevi.....	162
	Ripetizioni con percentuale.....	162
	Ripetizioni con tremolo.....	165
1.5	Note simultanee.....	167
1.5.1	Una voce.....	167
	Note in un accordo.....	167
	Ripetizione di un accordo.....	169
	Espressioni simultanee.....	171
	Cluster.....	172
1.5.2	Più voci.....	173
	Polifonia su un solo rigo.....	173
	Stili di voce.....	176
	Risoluzione delle collisioni.....	177
	Combinazione automatica delle parti.....	181
	Scrivere la musica in parallelo.....	187
1.6	Notazione del rigo.....	189
1.6.1	Aspetto del rigo.....	189
	Istanziare nuovi rigi.....	190
	Raggruppare i rigi.....	191
	Gruppi di rigi annidati.....	194
	Separare i sistemi.....	196
1.6.2	Modificare singoli rigi.....	197
	Simbolo del rigo.....	197
	Rigi ossia.....	200
	Nascondere i rigi.....	204
1.6.3	Scrittura delle parti.....	207

Nomi degli strumenti .....	207
Citare altre voci .....	211
Formattazione delle notine .....	214
1.7 Note editoriali .....	220
1.7.1 Interne al rigo .....	220
Scelta della dimensione del tipo di carattere .....	220
Indicazioni di diteggiatura .....	224
Note nascoste .....	226
Colorare gli oggetti .....	227
Parentesi .....	229
Gambi .....	229
1.7.2 Esterne al rigo .....	231
Nuvoletta di aiuto .....	231
Linee della griglia .....	232
Parentesi analitiche .....	234
1.8 Testo .....	235
1.8.1 Inserimento del testo .....	236
Scritte .....	236
Estensori del testo .....	237
Indicazioni testuali .....	239
Testo separato .....	241
1.8.2 Formattazione del testo .....	243
Introduzione al testo a margine .....	243
Scelta del tipo di carattere e della dimensione .....	245
Allineamento del testo .....	248
Notazione grafica nel blocco markup .....	251
Notazione musicale nel blocco markup .....	254
Testo formattato su più pagine .....	256
1.8.3 Tipi di carattere .....	257
Tipi di carattere in dettaglio .....	257
Tipi di carattere per singolo oggetto .....	259
Tipi di carattere per l'intero documento .....	259
<b>2 Notazione specialistica .....</b>	<b>261</b>
2.1 Musica vocale .....	261
2.1.1 Notazione comune per la musica vocale .....	261
Riferimenti per la musica vocale .....	261
Inserimento del testo vocale .....	262
Allineamento del testo alla melodia .....	263
Durate automatiche delle sillabe .....	265
Durate manuali delle sillabe .....	268
Più sillabe in una nota .....	269
Più note in una sillaba .....	270
Estensori e trattini .....	273
2.1.2 Tecniche specifiche per il testo vocale .....	273
Lavorare con testo e variabili .....	273
Posizionamento verticale del testo .....	275
Posizionamento orizzontale delle sillabe .....	279
Testo e ripetizioni .....	281
Testi alternati .....	289
Polifonia con testo in comune .....	290
2.1.3 Strofe .....	292
Aggiungere i numeri di strofa .....	292



Aggiungere le dinamiche alle strofe .....	293
Aggiungere i nomi dei cantanti alle strofe .....	293
Strofe con durate diverse .....	293
Stampare le strofe alla fine .....	296
Stampare le strofe alla fine in molteplici colonne .....	297
2.1.4 Canzoni .....	299
Riferimenti per canzoni .....	299
Canzonieri .....	300
2.1.5 Musica corale .....	300
Riferimenti per musica corale .....	300
Struttura di una partitura corale .....	301
Voci divise .....	302
2.1.6 Opera e musical .....	303
Riferimenti per opera e musical .....	304
Nomi dei personaggi .....	304
Suggerimenti musicali .....	306
Musica parlata .....	310
Dialogo sopra la musica .....	310
2.1.7 Canti salmi e inni .....	312
Riferimenti per canti e salmi .....	312
Impostare un canto .....	312
Salmi .....	319
Misure parziali nei motivi degli inni .....	322
2.1.8 Musica vocale antica .....	324
2.2 Keyboard and other multi-staff instruments .....	324
2.2.1 Common notation for keyboards .....	325
References for keyboards .....	325
Changing staff manually .....	326
Changing staff automatically .....	328
Staff-change lines .....	330
Cross-staff stems .....	330
2.2.2 Piano .....	332
Piano pedals .....	332
2.2.3 Accordion .....	333
Discant symbols .....	333
2.2.4 Harp .....	334
References for harps .....	334
Harp pedals .....	334
2.3 Unfretted string instruments .....	335
2.3.1 Common notation for unfretted strings .....	336
References for unfretted strings .....	336
Bowing indications .....	336
Harmonics .....	337
Snap (Bartók) pizzicato .....	338
2.4 Fretted string instruments .....	338
2.4.1 Common notation for fretted strings .....	339
References for fretted strings .....	339
String number indications .....	339
Default tablatures .....	341
Custom tablatures .....	356
Fret diagram markups .....	359
Predefined fret diagrams .....	369
Automatic fret diagrams .....	379
Right-hand fingerings .....	382

2.4.2	Guitar	383
	Indicating position and barring	384
	Indicating harmonics and dampened notes	384
	Indicating power chords	385
2.4.3	Banjo	387
	Banjo tablatures	387
2.5	Percussion	388
2.5.1	Common notation for percussion	388
	References for percussion	388
	Basic percussion notation	388
	Drum rolls	389
	Pitched percussion	390
	Percussion staves	390
	Custom percussion staves	392
	Ghost notes	396
2.6	Wind instruments	397
2.6.1	Common notation for wind instruments	397
	References for wind instruments	397
	Fingerings	398
2.6.2	Bagpipes	400
	Bagpipe definitions	400
	Bagpipe example	401
2.6.3	Woodwinds	402
	2.6.3.1 Woodwind diagrams	402
2.7	Chord notation	411
2.7.1	Chord mode	411
	Chord mode overview	411
	Common chords	412
	Extended and altered chords	413
2.7.2	Displaying chords	416
	Printing chord names	416
	Customizing chord names	418
2.7.3	Figured bass	424
	Introduction to figured bass	424
	Entering figured bass	425
	Displaying figured bass	428
2.8	Contemporary music	430
2.8.1	Pitch and harmony in contemporary music	430
	References for pitch and harmony in contemporary music	430
	Microtonal notation	430
	Contemporary key signatures and harmony	430
2.8.2	Contemporary approaches to rhythm	430
	References for contemporary approaches to rhythm	430
	Tuplets in contemporary music	431
	Contemporary time signatures	431
	Extended polymetric notation	431
	Beams in contemporary music	431
	Bar lines in contemporary music	431
2.8.3	Graphical notation	431
2.8.4	Contemporary scoring techniques	431
2.8.5	New instrumental techniques	431
2.8.6	Further reading and scores of interest	431
	Books and articles on contemporary musical notation	431
	Scores and musical examples	431

2.9	Ancient notation .....	431
2.9.1	Overview of the supported styles .....	433
2.9.2	Ancient notation—common features .....	433
	Pre-defined contexts .....	433
	Ligatures .....	434
	Custodes .....	434
2.9.3	Typesetting mensural music .....	435
	Mensural contexts .....	435
	Mensural clefs .....	436
	Mensural time signatures .....	437
	Mensural note heads .....	438
	Mensural flags .....	439
	Mensural rests .....	439
	Mensural accidentals and key signatures .....	440
	Annotational accidentals ( <i>musica ficta</i> ) .....	440
	White mensural ligatures .....	441
2.9.4	Typesetting Gregorian chant .....	442
	Gregorian chant contexts .....	442
	Gregorian clefs .....	443
	Gregorian accidentals and key signatures .....	444
	Divisiones .....	444
	Gregorian articulation signs .....	445
	Augmentum dots ( <i>morae</i> ) .....	446
	Gregorian square neume ligatures .....	446
2.9.5	Typesetting Kievan square notation .....	453
	Kievan contexts .....	453
	Kievan clefs .....	454
	Kievan notes .....	454
	Kievan accidentals .....	455
	Kievan bar line .....	455
	Kievan melismata .....	456
2.9.6	Working with ancient music—scenarios and solutions .....	457
	Incipits .....	457
	Mensurstriche layout .....	458
	Transcribing Gregorian chant .....	458
	Ancient and modern from one source .....	461
	Editorial markings .....	463
2.10	World music .....	463
2.10.1	Common notation for non-Western music .....	463
	Extending notation and tuning systems .....	463
2.10.2	Arabic music .....	464
	References for Arabic music .....	464
	Arabic note names .....	465
	Arabic key signatures .....	466
	Arabic time signatures .....	467
	Arabic music example .....	468
	Further reading for Arabic music .....	468
2.10.3	Turkish classical music .....	469
	References for Turkish classical music .....	469
	Turkish note names .....	469

<b>3</b>	<b>Input e output</b>	<b>471</b>
3.1	Struttura dell'input	471
3.1.1	Struttura di una partitura	471
3.1.2	Molteplici partiture in un libro	472
3.1.3	Molteplici file di output da un unico file di input	473
3.1.4	Nomi dei file di output	474
3.1.5	Struttura del file	475
3.2	Titoli e intestazioni	477
3.2.1	Creazione di titoli intestazioni e piè di pagina	477
	Come funzionano i titoli	477
	Formattazione predefinita dei titoli delle parti e dei brani	480
	Formattazione predefinita delle intestazioni e dei piè di pagina	483
3.2.2	Titoli intestazioni e piè di pagina personalizzati	484
	Titoli personalizzati	484
	Formattazione personalizzata dei titoli	485
	Formattazione personalizzata di intestazioni e piè di pagina	488
3.2.3	Creazione di metadati PDF	489
3.2.4	Creazione di note a piè di pagina	490
	Note a piè di pagina nelle espressioni musicali	490
	Note a piè di pagina nel testo separato	496
3.2.5	Riferimento ai numeri di pagina	498
3.2.6	Indice	499
3.3	Lavorare coi file di input	502
3.3.1	Inclusione di file LilyPond	502
3.3.2	Edizioni diverse da un unico sorgente	504
	Uso delle variabili	504
	Uso delle etichette	505
	Impostazioni globali	509
3.3.3	Caratteri speciali	509
	Codifica del testo	509
	Unicode	509
	Alias ASCII	510
3.4	Controllo dell'output	511
3.4.1	Estrarre frammenti musicali	511
3.4.2	Saltare la musica già corretta	512
3.4.3	Formati di output alternativi	513
3.4.4	Cambiare il tipo di carattere della notazione	513
3.5	Creazione dell'output MIDI	514
3.5.1	Notazione supportata nel MIDI	514
3.5.2	Notazione non supportata nel MIDI	515
3.5.3	Il blocco MIDI	515
3.5.4	Gestione delle dinamiche nel MIDI	515
	Dinamiche nel MIDI	516
	Impostazione del volume MIDI	516
	Impostazione delle proprietà del blocco MIDI	519
3.5.5	Uso degli strumenti MIDI	520
3.5.6	Uso delle ripetizioni nel MIDI	521
3.5.7	Mappatura dei canali MIDI	521
3.5.8	Proprietà di contesto per gli effetti MIDI	524
3.5.9	Miglioramento dell'output MIDI	525
	Lo script <code>articulate</code>	525
3.6	Estrazione dell'informazione musicale	525
3.6.1	Mostrare la notazione LilyPond	526

3.6.2	Mostrare le espressioni musicali scheme.....	526
3.6.3	Salvare eventi musicali in un file .....	526
<b>4</b>	<b>Gestione dello spazio .....</b>	<b>528</b>
4.1	Formattazione della pagina .....	528
4.1.1	Il blocco <code>\paper</code> .....	528
4.1.2	Formato carta e ridimensionamento automatico .....	529
	Impostare il formato carta .....	529
	Ridimensionamento automatico al formato carta .....	530
4.1.3	Variabili <code>\paper</code> della spaziatura verticale fissa .....	531
4.1.4	Variabili <code>\paper</code> della spaziatura verticale flessibile .....	531
	Struttura delle liste associative flessibili della spaziatura verticale .....	532
	Elenco delle variabili <code>\paper</code> flessibili della spaziatura verticale .....	533
4.1.5	Variabili <code>\paper</code> della spaziatura orizzontale .....	533
	Variabili <code>\paper</code> per larghezze e margini .....	534
	Variabili <code>\paper</code> per la modalità due pagine per foglio .....	535
	Variabili <code>\paper</code> per spostamenti e indentazioni .....	536
4.1.6	Altre variabili di <code>\paper</code> .....	536
	Variabili di <code>\paper</code> per l'interruzione di linea .....	536
	Variabili di <code>\paper</code> per l'interruzione di pagina .....	537
	Variabili di <code>\paper</code> per la numerazione delle pagine .....	538
	Svariate variabili di <code>\paper</code> .....	538
4.2	Formattazione della partitura .....	539
4.2.1	Il blocco <code>\layout</code> .....	539
4.2.2	Impostare la dimensione del rigo .....	541
4.3	Interruzioni .....	543
4.3.1	Interruzioni di linea .....	543
4.3.2	Interruzioni di pagina .....	546
	Interruzione di pagina manuale .....	546
	Interruzione di pagina ottimale .....	548
	Interruzione di pagina minimale .....	548
	Interruzione di pagina su una linea .....	548
	Voltata di pagina ottimale .....	548
4.4	Spaziatura verticale .....	549
4.4.1	Spaziatura verticale flessibile all'interno dei sistemi .....	549
	Proprietà della spaziatura dentro un sistema .....	550
	Spaziatura dei rigi non raggruppati .....	553
	Spaziatura dei rigi raggruppati .....	554
	Spaziatura delle linee che non sono rigi .....	555
4.4.2	Posizionamento esplicito di rigi e sistemi .....	556
4.4.3	Elusione delle collisioni verticali .....	563
4.5	Spaziatura orizzontale .....	564
4.5.1	Panoramica sulla spaziatura orizzontale .....	564
4.5.2	Nuova spaziatura nel corso di un brano .....	566
4.5.3	Modifica della spaziatura orizzontale .....	567
4.5.4	Larghezza della linea .....	569
4.5.5	Notazione proporzionale .....	569
4.6	Riduzione del numero di pagine di una partitura .....	575
4.6.1	Visualizzare la spaziatura .....	576
4.6.2	Modificare la spaziatura .....	577

<b>5</b>	<b>Modifica delle impostazioni predefinite</b>	<b>579</b>
5.1	Contesti di interpretazione	579
5.1.1	Contexts explained	579
	Output definitions - blueprints for contexts	579
	Score - the master of all contexts	580
	Top-level contexts - staff containers	580
	Intermediate-level contexts - staves	580
	Bottom-level contexts - voices	581
5.1.2	Creating and referencing contexts	581
5.1.3	Keeping contexts alive	584
5.1.4	Modifying context plug-ins	587
5.1.5	Changing context default settings	589
	Changing all contexts of the same type	589
	Changing just one specific context	592
	Order of precedence	593
5.1.6	Defining new contexts	594
5.1.7	Context layout order	596
5.2	Come funziona la Guida al funzionamento interno	598
5.2.1	Navigating the program reference	598
5.2.2	Layout interfaces	599
5.2.3	Determining the grob property	600
5.2.4	Naming conventions	601
5.3	Modifica delle proprietà	601
5.3.1	Overview of modifying properties	601
5.3.2	The <code>\set</code> command	601
5.3.3	The <code>\override</code> command	603
5.3.4	Il comando <code>\tweak</code>	605
5.3.5	<code>\set</code> vs. <code>\override</code>	607
5.3.6	Modifying alists	608
5.4	Proprietà e concetti utili	610
5.4.1	Input modes	610
5.4.2	Direction and placement	611
	Articulation direction indicators	611
	The direction property	611
5.4.3	Distances and measurements	612
5.4.4	Dimensions	613
5.4.5	Staff symbol properties	613
5.4.6	Spanners	614
	Using the <code>spanner-interface</code>	614
	Using the <code>line-spanner-interface</code>	616
5.4.7	Visibility of objects	618
	Removing the stencil	619
	Making objects transparent	619
	Painting objects white	620
	Using break-visibility	620
	Special considerations	622
5.4.8	Line styles	625
5.4.9	Rotating objects	625
	Rotating layout objects	626
	Rotating markup	626
5.5	Ritocchi avanzati	626
5.5.1	Aligning objects	627
	Setting <code>X-offset</code> and <code>Y-offset</code> directly	627

Using the <code>side-position-interface</code> .....	628
Using the <code>self-alignment-interface</code> .....	628
Using the <code>break-alignable-interface</code> .....	629
5.5.2 Vertical grouping of grobs.....	631
5.5.3 Modifying stencils.....	631
5.5.4 Modifying shapes.....	632
Modifying ties and slurs.....	632
5.5.5 Modifying broken spanners.....	636
Using <code>\alterBroken</code> .....	636
5.5.6 Unpure-pure containers.....	638
5.6 Uso delle funzioni musicali.....	639
5.6.1 Substitution function syntax.....	639
5.6.2 Substitution function examples.....	640

## Appendice A Tabelle del manuale della notazione.....643

A.1 Grafico dei nomi degli accordi.....	643
A.2 Modificatori degli accordi.....	644
A.3 Accordature predefinite.....	647
A.4 Diagrammi degli accordi predefiniti.....	649
Diagrammi per chitarra.....	649
Diagrammi per ukulele.....	650
Diagrammi per mandolino.....	652
A.5 Formati carta predefiniti.....	654
A.6 Strumenti MIDI.....	658
A.7 Elenco dei colori.....	658
A.8 Il tipo di carattere Feta.....	660
Glifi della chiave.....	660
Glifi delle indicazioni di tempo.....	661
Glifi dei numeri.....	661
Glifi delle alterazioni.....	661
Glifi delle teste di nota predefinite.....	662
Glifi delle teste di nota speciali.....	663
Glifi delle teste di nota a forma variabile.....	663
Glifi delle pause.....	667
Glifi delle code.....	668
Glifi dei punti.....	668
Glifi delle dinamiche.....	669
Glifi dei segni.....	669
Glifi delle teste a forma di freccia.....	671
Glifi delle estremità delle parentesi.....	672
Glifi dei pedali.....	672
Glifi della fisarmonica.....	672
Glifi delle legature di valore.....	672
Glifi della notazione vaticana.....	673
Glifi della notazione medicea.....	674
Glifi Hufnagel.....	674
Glifi della notazione mensurale.....	675
Glifi della notazione neomensurale.....	678
Glifi Petrucci.....	679
Glifi Solesmes.....	680
Glifi della notazione di Kiev.....	680
A.9 Stili delle teste di nota.....	681
A.10 Stili della chiave.....	682

A.11	Comandi per <i>markup</i> .....	683
A.11.1	Font .....	683
A.11.2	Align .....	693
A.11.3	Graphic .....	708
A.11.4	Music .....	716
A.11.5	Instrument Specific Markup .....	722
A.11.6	Accordion Registers .....	725
A.11.7	Other .....	730
A.12	Comandi per una lista di <i>markup</i> .....	737
A.13	Elenco dei caratteri speciali .....	738
A.14	Elenco delle articolazioni .....	740
	Articolazioni .....	740
	Ornamenti .....	740
	Punti coronati .....	740
	Segni specifici per strumento .....	741
	Segni di ripetizione .....	741
	Segni antichi .....	741
A.15	Note percussive .....	742
A.16	Glossario tecnico .....	744
	alist .....	744
	callback .....	744
	closure .....	744
	glyph .....	745
	grob .....	745
	immutable .....	745
	interface .....	745
	lexer .....	746
	mutable .....	746
	output-def .....	746
	parser .....	746
	parser variable .....	746
	prob .....	747
	smob .....	747
	stencil .....	747
A.17	Tutte le proprietà di contesto .....	747
A.18	Proprietà della formattazione .....	760
A.19	Funzioni musicali disponibili .....	781
A.20	Identificatori delle modifiche di contesto .....	791
A.21	Tipi di predicati predefiniti .....	792
	R5RS primary predicates .....	792
	R5RS secondary predicates .....	792
	Guile predicates .....	793
	LilyPond scheme predicates .....	793
	LilyPond exported predicates .....	793
A.22	Funzioni Scheme .....	794
<b>Appendice B Schema riassuntivo .....</b>		<b>819</b>
<b>Appendice C GNU Free Documentation License .....</b>		<b>823</b>
<b>Appendice D Indice dei comandi di LilyPond .....</b>		<b>830</b>



<b>Appendice E</b>	<b>Indice di LilyPond .....</b>	<b>841</b>
--------------------	---------------------------------	------------

# 1 Notazione musicale

Questo capitolo spiega come creare la notazione musicale.

## 1.1 Altezze

*dolce e molto legato*

*p*

*cresc.*

*sf*

Red. \*

Red. \*

Red. \*

Red. \*

38

*p*

Red. \*

Questa sezione tratta il modo in cui si determina l'altezza delle note. Occorre considerare tre aspetti: input, modifica e output.

### 1.1.1 Inserimento delle altezze

Questa sezione spiega come indicare l'altezza delle note. Ci sono due modi di collocare le note in una determinata ottava: il modo assoluto e il modo relativo. Nella maggioranza dei casi il modo relativo è più funzionale.

#### Ottava assoluta

Le altezze, se non si adotta una lingua diversa, sono scritte in notazione olandese, che usa le lettere minuscole dalla a (La) alla g (Sol). Le note c (Do) e b (Si) vengono scritte un'ottava sotto il Do centrale.

```
{
  \clef bass
  c4 d e f
  g4 a b c
  d4 e f g
}
```

Si possono indicare altre ottave con l'apice singolo (') o la virgola (,). Ogni ' alza l'altezza di un'ottava; ogni , abbassa l'altezza di un'ottava.

```
{
  \clef treble
  c'4 e' g' c''
  c'4 g b c'
  \clef bass
  c,4 e, g, c
  c,4 g,, b,, c,
}
```



I normali segni di ottava possono essere inseriti una sola volta se si imposta un'altezza di riferimento dopo `\fixed` e prima della musica. Le altezze inserite in un blocco `\fixed` hanno bisogno dei segni ' o , solo quando si trovano sopra o sotto l'ottava dell'altezza di riferimento.

```
{
  \fixed c' {
    \clef treble
    c4 e g c'
    c4 g, b, c
  }
  \clef bass
  \fixed c, {
    c4 e g c'
    c4 g, b, c
  }
}
```



Le altezze dell'espressione musicale che segue `\fixed` non cambiano se racchiuse da un blocco `\relative`, che vedremo tra poco.

## Vedi anche

Glossario musicale: Sezione “Nomi delle altezze” in *Glossario Musicale*.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

## Ottava relativa

L'inserimento delle note con l'ottava assoluta costringe a specificare l'ottava di ogni singola nota. Al contrario, se si usa l'ottava relativa, ogni ottava è determinata dall'ultima nota: se si cambia l'ottava di una nota, cambieranno anche le ottave di tutte le note successive.

La modalità relativa deve essere impostata in modo esplicito col comando `\relative`:

```
\relative altezza_di_riferimento espressione_musicale
```

In modalità relativa ogni nota è collocata il più vicino possibile a quella precedente. Questo significa che l'ottava di ogni altezza all'interno di *espressione\_musicale* viene calcolata nel modo seguente:

- In assenza di segni di cambiamento d'ottava, l'ottava di un'altezza viene calcolata in modo che l'intervallo con la nota precedente sia inferiore a una quinta. Tale intervallo è determinato senza considerare gli accidenti.
- Si può aggiungere un segno di cambiamento d'ottava ' o , per collocare l'altezza di una nota all'ottava superiore o inferiore a quella di riferimento.
- È possibile usare più di un segno di cambiamento d'ottava. Per esempio, '' e ,, modificano l'altezza di due ottave.
- L'altezza della prima nota è relativa a *altezza\_di\_riferimento*. *altezza\_di\_riferimento* è specificato nel modo di ottava assoluta. Quale di queste opzioni è la più conveniente?

un'ottava del c

Identificare il Do centrale con c' è molto semplice, quindi trovare le ottave del c (Do) sarà altrettanto semplice. Se la musica inizia con gis sopra c'', si scriverà qualcosa simile a `\relative { gis'' ... }`

un'ottava della prima nota

Scrivere `\relative { gis'' ... }` è un modo semplice per determinare l'altezza assoluta della prima nota dell'espressione musicale.

nessuna altezza di riferimento esplicita

La forma `\relative {gis'' ... }` è una versione compatta dell'opzione precedente: la prima nota dentro l'espressione musicale è scritta come altezza assoluta. In questo caso equivale a scegliere f come altezza di riferimento.

La documentazione di solito usa l'ultima opzione.

Ecco il modo relativo in azione:

```
\relative {
  \clef bass
  c d e f
  g a b c
  d e f g
}
```



I segni di cambiamento d'ottava si impiegano per gli intervalli più ampi di quello di quarta:

```
\relative {
  c'' g c f,
  c' a, e'' c
}
```



Una sequenza di note senza segni di ottava può tuttavia comprendere intervalli di grande estensione:

```
\relative {
  c f b e
  a d g c
}
```



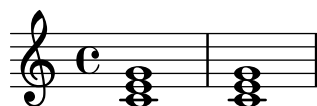
Nel caso di blocchi `\relative` annidati, il blocco `\relative` più interno inizia con la propria altezza di riferimento, indipendentemente dal `\relative` più esterno.

```
\relative {
  c' d e f
  \relative {
    c'' d e f
  }
}
```



`\relative` non ha effetto sui blocchi `\chordmode`.

```
\new Staff {
  \relative c''' {
    \chordmode { c1 }
  }
  \chordmode { c1 }
}
```



`\relative` non può essere inserito all'interno dei blocchi `\chordmode`.

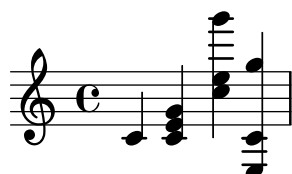
La musica all'interno di un blocco `\transpose` è considerata in notazione d'ottava assoluta, a meno che non sia incluso il blocco `\relative`.

```
\relative {
  d' e
  \transpose f g {
    d e
    \relative {
      d' e
    }
  }
}
```



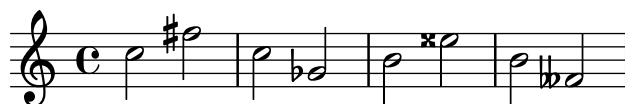
Se l'elemento precedente è un accordo, il posizionamento dell'ottava della nota o dell'accordo che segue è riferito alla prima nota dell'accordo stesso. All'interno degli accordi la nota successiva è sempre relativa a quella precedente. Esaminate con attenzione l'esempio seguente, e in particolare le note c.

```
\relative {
  c'
  <c e g>
  <c' e g'>
  <c, e, g'>
}
```



Come spiegato sopra, il riferimento delle altezze a un'ottava è calcolato in base ai soli nomi delle note, senza considerare le alterazioni. Dunque un Mi doppio diesis che segue un Si verrà posizionato sopra, mentre un Fa doppio bemolle sarà posizionato sotto. In altre parole, un intervallo di quarta aumentata due volte viene considerato più piccolo di una quinta diminuita due volte, indipendentemente dal numero di semitoni contenuto in ogni intervallo.

```
\relative {
  c''2 fis
  c2 ges
  b2 eisis
  b2 feses
}
```



## Vedi anche

Glossario musicale: Sezione “quinta” in *Glossario Musicale*, Sezione “intervallo” in *Glossario Musicale*, Sezione “Nomi delle altezze” in *Glossario Musicale*.

Guida alla notazione: [\[Octave checks\]](#), pagina [\[undefined\]](#).

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “RelativeOctaveMusic” in *Guida al Funzionamento Interno*.

## Alterazioni

**Nota:** I nuovi utenti sono talvolta confusi dalla gestione delle alterazioni e delle armature di chiave. In LilyPond i nomi delle note specificano le altezze; le armature e le chiavi determinano come queste altezze debbano essere rappresentate. Una nota non alterata come `c` significa ‘Do naturale’, indipendentemente dall’armatura o dalla chiave. Per maggiori informazioni si veda Sezione “Altezze e armature di chiave” in *Manuale di Apprendimento*.

Nella modalità di notazione predefinita un *diesis* si ottiene aggiungendo `is` al nome della nota, un *bemolle* aggiungendo `es`. Come potete immaginare, un *doppio diesis* o *doppio bemolle* si ottengono aggiungendo `isis` o `eses`. Questa sintassi è desunta dalla notazione olandese. Per usare altri nomi per le alterazioni, si veda [\[Note names in other languages\]](#), pagina [\[undefined\]](#).

```
\relative c'' { ais1 aes aisis aeses }
```



Un’altezza naturale è indicata con il semplice nome della nota; non è richiesto alcun suffisso. Un segno di bequadro apparirà automaticamente quando occorre cancellare l’armatura di chiave o l’effetto di un’alterazione precedente.

```
\relative c'' { a4 aes a2 }
```



È possibile indicare alterazioni di quarti di tono. Ecco una serie di Do con altezza crescente:

```
\relative c'' { ceseh1 ces ceh c cih cis cisih }
```



Di norma le alterazioni vengono mostrate automaticamente, ma è possibile anche inserirle manualmente. Si può forzare l’inserimento di un’alterazione di sicurezza aggiungendo il punto esclamativo `!` dopo l’altezza. Un’alterazione di cortesia (ovvero un’alterazione compresa tra parentesi) si ottiene aggiungendo il punto interrogativo `?` dopo l’altezza.

```
\relative c'' { cis cis cis! cis? c c c! c? }
```



Se una nota è prolungata attraverso una legatura di valore, l’alterazione viene ripetuta solo all’inizio di un nuovo sistema:

```
\relative c'' {
  cis1~ 1~
  \break
  cis
}
```

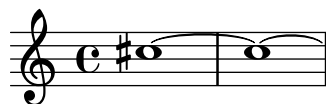


## Frammenti di codice selezionati

*Nascondere le alterazioni delle note con legatura di valore all'inizio di un nuovo sistema*

Questo frammento mostra come nascondere le alterazioni delle note unite alla figura precedente mediante una legatura di valore all'inizio di un nuovo sistema

```
\relative c'' {
  \override Accidental.hide-tied-accidental-after-break = ##t
  cis1~ cis~
  \break
  cis
}
```



*Impedire l'inserimento automatico dei bequadri supplementari*

Secondo le norme tipografiche tradizionali, un segno di bequadro viene inserito prima di un diesis o di un bemolle se un precedente doppio diesis o bemolle sulla stessa nota è cancellato. Per cambiare questo comportamento e seguire la pratica contemporanea, si imposta la proprietà `extraNatural` su `f` (falso) nel contesto `Staff`.

```
\relative c'' {
  aeses4 aes ais a
  \set Staff.extraNatural = ##f
  aeses4 aes ais a
}
```



## Vedi anche

Glossario musicale: Sezione “diesis” in *Glossario Musicale*, Sezione “bemolle” in *Glossario Musicale*, Sezione “doppio diesis” in *Glossario Musicale*, Sezione “doppio bemolle” in *Glossario Musicale*, Sezione “Nomi delle altezze” in *Glossario Musicale*, Sezione “quarto di tono” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Altezze e armature di chiave” in *Manuale di Apprendimento*.



Guida alla notazione: `\undefined` [Automatic accidentals], pagina `\undefined`, [Annotational accidentals (musica ficta)], pagina 440, `\undefined` [Note names in other languages], pagina `\undefined`.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Accidental-engraver” in *Guida al Funzionamento Interno*, Sezione “Accidental” in *Guida al Funzionamento Interno*, Sezione “AccidentalCautionary” in *Guida al Funzionamento Interno*, Sezione “accidental-interface” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Poiché non esistono standard universalmente accettati per indicare le alterazioni di quarto di tono, il simbolo impiegato da LilyPond non si riferisce ad alcuno standard.

## Nomi delle note in altre lingue

Lilypond comprende insieme predefiniti di nomi di note e alterazioni in altre lingue. La scelta della lingua si fa solitamente all’inizio del file; l’esempio seguente è scritto in notazione italiana:

```
\language "italiano"
```

```
\relative {
  do' re mi sib
}
```



Le lingue disponibili e i tipi di notazione che definiscono sono:

Lingua	Nomi delle note
nederlands	c d e f g a bes b
catalan	do re mi fa sol la sib si
deutsch	c d e f g a b h
english	c d e f g a bf b
espanol o	do re mi fa sol la sib si
español	
italiano o	do re mi fa sol la sib si
français	
norsk	c d e f g a b h
portugues	do re mi fa sol la sib si
suomi	c d e f g a b h
svenska	c d e f g a b h
vlaams	do re mi fa sol la sib si

Oltre ai nomi delle note, anche i suffissi per le alterazioni possono variare a seconda della lingua adottata:

Lingua	diesis	bemolle	doppio diesis	doppio bemolle
nederlands	-is	-es	-isis	-eses
catalan	-d/-s	-b	-dd/-ss	-bb
deutsch	-is	-es	-isis	-eses
english	-s/-sharp	-f/-flat	-ss/-x/-sharpsharp	-ff/-flatflat

espanol	o	-s	-b	-ss/-x	-bb
español					
italiano	o	-d	-b	-dd	-bb
français					
norsk		-iss/-is	-ess/-es	-ississ/-isis	-essess/-eses
portugues		-s	-b	-ss	-bb
suomi		-is	-es	-isis	-eses
svenska		-iss	-ess	-ississ	-essess
vlaams		-k	-b	-kk	-bb

In olandese, *aes* viene contratto in *as*, ma entrambe le forme sono accettate in LilyPond. Analogamente, sia *es* che *ees* sono accettati. Lo stesso vale per *aeses* / *ases* e *eeses* / *eses*. Talvolta solo questi nomi contratti sono definiti nei corrispondenti file della lingua.

```
\relative c'' { a2 as e es a ases e eses }
```



In alcune forme musicali vengono usati i microtoni, le cui alterazioni sono frazioni di un ‘normale’ diesis o bemolle. La seguente tabella elenca i nomi delle note per le alterazioni di un quarto di tono in varie lingue; i prefissi *semi-* e *sesqui-* significano rispettivamente ‘metà’ e ‘uno e mezzo’. Le lingue che non compaiono in questa tabella non hanno ancora dei nomi per le note speciali.

Lingua	semi-diesis	semi-bemolle	sesqui-diesis	sesqui-bemolle
nederlands	-ih	-eh	-isih	-esch
deutsch	-ih	-eh	-isih	-esch
english	-qs	-qf	-tqs	-tqf
espanol	o -cs	-cb	-tcs	-tcb
español				
italiano	o -sd	-sb	-dsd	-bsb
français				
portugues	-sqf	-bqt	-stqt	-btqt

Gran parte delle lingue presentate qui sono comunemente associate alla musica classica occidentale, nota anche come *Common Practice Period*. Sono tuttavia supportati anche altezze e sistemi di accordatura alternativi: si veda Sezione 2.10.1 [Common notation for non-Western music], pagina 463.

## Vedi anche

Glossario musicale: Sezione “Nomi delle altezze” in *Glossario Musicale*, Sezione “Periodo di pratica comune” in *Glossario Musicale*.

Guida alla notazione: Sezione 2.10.1 [Common notation for non-Western music], pagina 463.

File installati: `scm/define-note-names.scm`.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

### 1.1.2 Modifica di più altezze

Questa sezione tratta il modo di modificare le altezze delle note.

## Controlli di ottava

In modalità relativa è facile dimenticare un segno di cambiamento d'ottava. I controlli di ottava permettono di rilevare questi errori più facilmente: infatti, generano un avviso e correggono l'ottava se una nota si trova in un'ottava diversa dal previsto.

Per controllare l'ottava di una nota, occorre specificare l'ottava assoluta dopo il simbolo `=`. Questo esempio genererà un avviso (e cambierà l'altezza) perché la seconda nota è l'ottava assoluta `d''` invece di `d'`, come indicato dalla correzione di ottava.

```
\relative {
  c''2 d='4 d
  e2 f
}
```



L'ottava in cui si trovano le note può essere controllata anche col comando `\octaveCheck altezza_di_controllo`. L'`altezza_di_controllo` è specificata in modo assoluto. Questo comando controlla che l'intervallo tra la nota precedente e l'`altezza_di_controllo` sia compresa in una quinta (ovvero secondo il normale calcolo della modalità relativa). Se il controllo fallisce, compare un avviso. Benché la nota precedente non sia modificata, le note successive sono relative al valore corretto.

```
\relative {
  c''2 d
  \octaveCheck c'
  e2 f
}
```



Nelle due battute che seguono, il primo e il terzo `\octaveCheck` falliscono, mentre il secondo non fallisce.

```
\relative {
  c''4 f g f

  c4
  \octaveCheck c'
  f
  \octaveCheck c'
  g
  \octaveCheck c'
  f
}
```



## Vedi anche

Frammenti di codice: Sezione “Pitches” in *snippets*.

Guida al funzionamento interno: Sezione “RelativeOctaveCheck” in *Guida al Funzionamento Interno*.

## Trasposizione

Un’espressione musicale può essere trasposta con `\transpose`. La sintassi è

```
\transpose altezza_di_partenza altezza_di_arrivo espressione_musicale
```

Significa che *espressione\_musicale* viene trasposto dell’intervallo compreso tra le altezze *altezza\_di\_partenza* e *altezza\_di\_arrivo*: qualsiasi nota che presenti un’altezza corrispondente all’*altezza\_di\_partenza* viene modificata in *altezza\_di\_arrivo*, e qualsiasi altra nota viene trasposta dello stesso intervallo. Entrambe le altezze sono inserite in modalità assoluta.

**Nota:** La musica all’interno di un blocco `\transpose` è assoluta a meno che il blocco non includa un `\relative`.

Prendiamo come esempio un brano scritto in Re maggiore. Possiamo trasportarlo in Mi maggiore; si noti come anche l’armatura di chiave venga trasposta automaticamente.

```
\transpose d e {
  \relative {
    \key d \major
    d'4 fis a d
  }
}
```



Se una parte scritta in Do (l’*intonazione reale* abituale) deve essere suonata su un clarinetto in La (per il quale un La viene rappresentato da un Do e dunque suona una terza minore più basso), la trasposizione sarà ottenuta con:

```
\transpose a c' {
  \relative {
    \key c \major
    c'4 d e g
  }
}
```



Si noti che `\key c \major` è specificato esplicitamente. Se non si specifica un’armatura di chiave, le note verranno trasposte ma non apparirà alcuna armatura.

`\transpose` fa distinzione tra altezze enarmoniche: sia `\transpose c cis` che `\transpose c des` traspongono un brano di un semitono più alto. La prima versione mostrerà i diesis e le note rimarranno sullo stesso grado della scala, mentre la seconda versione mostrerà i bemolli sul grado superiore della scala.

```
music = \relative { c' d e f }
```

```
\new Staff {
  \transpose c cis { \music }
  \transpose c des { \music }
}
```



`\transpose` può essere usato anche in un altro modo, ovvero per inserire note scritte per uno strumento traspositore. Gli esempi precedenti mostrano come inserire altezze in Do (o *intonazione reale*) e mostrare le note di uno strumento traspositore, ma è possibile anche il contrario: per esempio, se da un insieme di parti strumentali si volesse ricavare una partitura per il direttore. Così, per inserire la parte per una tromba in Si bemolle che inizia con un Mi (intonazione reale Re), si può scrivere:

```
musicInBflat = { e4 ... }
\transpose c bes, \musicInBflat
```

Per stampare questa musica in Fa (ad esempio per riarrangiarla per corno) si può avvolgere la musica esistente in un altro `\transpose`:

```
musicInBflat = { e4 ... }
\transpose f c' { \transpose c bes, \musicInBflat }
```

Per maggiori informazioni sugli strumenti traspositori, si veda [\[Instrument transpositions\]](#), pagina [\[undefined\]](#).

## Frammenti di codice selezionati

### *Trasposizione delle altezze con numero minimo di alterazioni*

Questo esempio usa del codice Scheme per imporre delle modifiche enarmoniche alle note che permettano di avere il numero minimo di alterazioni. In questo caso si applica la seguente regola:

Le doppie alterazioni devono essere eliminate

Si diesis -> Do

Mi diesis -> Fa

Do bemolle -> Si

Fa bemolle -> Mi

In questo modo vengono scelti i suoni enarmonici più semplici.

```
#(define (naturalize-pitch p)
  (let ((o (ly:pitch-octave p))
        (a (* 4 (ly:pitch-alteration p)))
        ;; alteration, a, in quarter tone steps,
        ;; for historical reasons
        (n (ly:pitch-notename p)))
    (cond
      ((and (> a 1) (or (eq? n 6) (eq? n 2))))
      (set! a (- a 2))
      (set! n (+ n 1)))
    ((and (< a -1) (or (eq? n 0) (eq? n 3))))
    (set! a (+ a 2))
    (set! n (- n 1))))
  (cond
```

```

    ((> a 2) (set! a (- a 4)) (set! n (+ n 1)))
    ((< a -2) (set! a (+ a 4)) (set! n (- n 1)))
    (if (< n 0) (begin (set! o (- o 1)) (set! n (+ n 7))))
    (if (> n 6) (begin (set! o (+ o 1)) (set! n (- n 7))))
    (ly:make-pitch o n (/ a 4)))

#(define (naturalize music)
  (let ((es (ly:music-property music 'elements))
        (e (ly:music-property music 'element))
        (p (ly:music-property music 'pitch)))
    (if (pair? es)
        (ly:music-set-property!
         music 'elements
         (map (lambda (x) (naturalize x)) es)))
    (if (ly:music? e)
        (ly:music-set-property!
         music 'element
         (naturalize e)))
    (if (ly:pitch? p)
        (begin
         (set! p (naturalize-pitch p))
         (ly:music-set-property! music 'pitch p)))
    music))

naturalizeMusic =
#(define-music-function (m)
  (ly:music?)
  (naturalize m))

music = \relative c' { c4 d e g }

\score {
  \new Staff {
    \transpose c ais { \music }
    \naturalizeMusic \transpose c ais { \music }
    \transpose c deses { \music }
    \naturalizeMusic \transpose c deses { \music }
  }
  \layout { }
}

```



## Vedi anche

Guida alla notazione: [\[Instrument transpositions\]](#), pagina [\[Inversion\]](#), pagina [\[Modal transformations\]](#), pagina [\[Relative octave entry\]](#), pagina [\[Retrograde\]](#), pagina [\[Retrograde\]](#).

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TransposedMusic” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

La conversione relativa non avrà effetto sulle sezioni `\transpose`, `\chordmode` e `\relative` comprese all’interno di un blocco `\relative`. Per usare la modalità relativa all’interno di musica trasposta, occorre inserire un ulteriore blocco `\relative` all’interno di `\transpose`.

Il comando `\transpose` impedisce di stampare le alterazioni triple. Le sostituisce con un’altezza ‘enarmonicamente equivalente’ (per esempio, Re bemolle al posto di Mi triplo bemolle).

## Inversione

Un’espressione musicale può essere invertita e trasposta in una singola operazione con:

```
\inversion altezza-di-riferimento altezza-di-arrivo espressione_musicale
```

L’*espressione\_musicale* viene invertita intervallo per intervallo intorno all’*altezza-di-riferimento* e poi trasposta in modo che ci sia una corrispondenza tra *altezza-di-riferimento* e *altezza-di-arrivo*.

```
music = \relative { c' d e f }
\new Staff {
  \music
  \inversion d' d' \music
  \inversion d' ees' \music
}
```



**Nota:** I motivi da invertire devono essere scritti in forma assoluta oppure devono essere prima convertiti in forma assoluta racchiudendoli in un blocco `\relative`.

## Vedi anche

Guida alla notazione: [\[Modal transformations\]](#), pagina [\[Retrograde\]](#), pagina [\[Transpose\]](#), pagina [\[TransposedMusic\]](#).

## Retrogradazione

Un’espressione musicale può essere invertita in modo da produrre il proprio retrogrado:

```
music = \relative { c'8. ees16( fis8. a16 b8.) gis16 f8. d16 }
\new Staff {
  \music
  \retrograde \music
}
```



## Problemi noti e avvertimenti

Le legature di valore manuali in `\retrograde` saranno spezzate e genereranno degli avvisi. Alcune legature di valore possono essere generate automaticamente abilitando `\Automatic note splitting`, pagina [\(undefined\)](#).

## Vedi anche

Guida alla notazione: [\(undefined\)](#) [Inversion], pagina [\(undefined\)](#), [\(undefined\)](#) [Modal transformations], pagina [\(undefined\)](#), [\(undefined\)](#) [Transpose], pagina [\(undefined\)](#).

## Trasposizioni modali

In una composizione musicale basata su una scala, un motivo viene frequentemente trasportato in differenti modi. Può essere *trasposto* per iniziare in punti diversi della scala o può essere *invertito* rispetto a un punto cardine della scala. Può anche essere rovesciato per produrre il *retrogrado*, si veda [\(undefined\)](#) [Retrograde], pagina [\(undefined\)](#).

**Nota:** Le note che non si trovano all'interno della scala definita non vengono trasformate.

## Trasposizione modale

Un motivo può essere trasposto entro una certa scala con:

```
\modalTranspose altezza-di-partenza altezza-di-arrivo scala motif
```

Le note di *motif* vengono spostate, se all'interno della *scala*, del numero di gradi della scala dati dall'intervallo tra *altezza-di-arrivo* e *altezza-di-partenza*:

```
diatonicScale = \relative { c' d e f g a b }
motif = \relative { c'8 d e f g a b c }
```

```
\new Staff {
  \motif
  \modalTranspose c f \diatonicScale \motif
  \modalTranspose c b, \diatonicScale \motif
}
```



È possibile indicare una scala ascendente di qualsiasi lunghezza e con qualsiasi intervallo:

```
pentatonicScale = \relative { ges aes bes des ees }
motif = \relative { ees'8 des ges,4 <ges' bes,> <ges bes,> }
```

```
\new Staff {
  \motif
  \modalTranspose ges ees' \pentatonicScale \motif
}
```





Se usato con una scala cromatica, `\modalTranspose` ha un effetto simile a `\transpose`, con in più la possibilità di specificare i nomi delle note da usare:

```
chromaticScale = \relative { c' cis d dis e f fis g gis a ais b }
motif = \relative { c'8 d e f g a b c }
```

```
\new Staff {
  \motif
  \transpose c f \motif
  \modalTranspose c f \chromaticScale \motif
}
```



### *Inversione modale*

Una sequenza di note può essere invertita all'interno di una data scala intorno a una determinata nota cardine e quindi trasposto, in un'unica operazione, con:

```
\modalInversion altezza-cardine altezza-di-arrivo scala motif
```

Le note di *motif* vengono spostate dello stesso numero di gradi dalla nota dell'*altezza-cardine* all'interno della *scala*, ma nella direzione opposta, e il risultato viene poi spostato all'interno della *scala* per il numero di gradi dato dall'intervallo tra *altezza-di-arrivo* e *altezza-cardine*.

Dunque, per invertire intorno a una particolare nota della scala, è necessario usare il medesimo valore per *altezza-cardine* e *altezza-di-arrivo*:

```
octatonicScale = \relative { ees' f fis gis a b c d }
motif = \relative { c'8. ees16 fis8. a16 b8. gis16 f8. d16 }
```

```
\new Staff {
  \motif
  \modalInversion fis' fis' \octatonicScale \motif
}
```



Per invertire intorno a una nota cardine posta tra altre due note, si inverte intorno a una della note e poi si traspone di un grado della scala. Le due note specificate possono essere interpretate come parentesi del punto cardine:

```
scale = \relative { c' g' }
motive = \relative { c' c g' c, }
```

```
\new Staff {
  \motive
  \modalInversion c' g' \scale \motive
}
```



L'operazione combinata di inversione e retrogradazione produce la retrogradazione inversa:

```
octatonicScale = \relative { ees' f fis gis a b c d }
motif = \relative { c'8. ees16 fis8. a16 b8. gis16 f8. d16 }

\new Staff {
  \motif
  \retrograde \modalInversion c' c' \octatonicScale \motif
}
```



## Vedi anche

Guida alla notazione: [\[Inversion\]](#), pagina [\[Retrograde\]](#), pagina [\[Transpose\]](#), pagina [\[Transpose\]](#).

### 1.1.3 Aspetto delle altezze

Questa sezione tratta il modo di modificare l'aspetto delle altezze delle note.

## Chiave

Senza un comando esplicito, la chiave predefinita in LilyPond è la chiave di violino (o di *Sol*).

```
c'2 c'
```



Per cambiare la chiave si usa il comando `\clef` seguito dal nome della chiave. In tutti gli esempi seguenti viene mostrato il *Do centrale*.

```
\clef treble
c'2 c'
\clef alto
c'2 c'
\clef tenor
c'2 c'
\clef bass
c'2 c'
```



L'elenco completo di tutti i nomi di chiave possibili si trova in [\[Clef styles\]](#), pagina [\[Mensural clefs\]](#), pagina 436, e [\[Gregorian clefs\]](#), pagina 443. La musica che utilizza le chiavi dell'intavolatura è discussa in [\[Default tablatures\]](#), pagina 341, e [\[Custom tablatures\]](#), pagina 356.

Per mischiare le chiavi quando si usano le notine, leggere come si usano i comandi `\cueClef` e `\cueDuringWithClef` descritti in [\[Formatting cue notes\]](#), pagina [\[Formatting cue notes\]](#).

Aggiungendo `_8` o `^8` al nome della chiave, la sua adozione comporta il trasporto all'ottava rispettivamente inferiore o superiore, mentre `_15` e `^15` traspongono di due ottave. È possibile usare altri numeri interi, se necessario. I nomi di chiave contenenti caratteri non alfabetici devono essere racchiusi tra virgolette

```
\clef treble
c'2 c'
\clef "treble_8"
c'2 c'
\clef "bass^15"
c'2 c'
\clef "alto_2"
c'2 c'
\clef "G_8"
c'2 c'
\clef "F^5"
c'2 c'
```



L'ottavazione opzionale si può ottenere racchiudendo l'argomento numerico tra parentesi tonde o quadre:

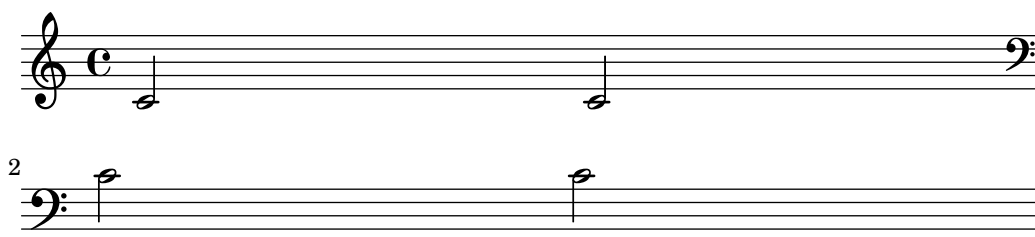
```
\clef "treble_(8)"
c'2 c'
\clef "bass^[15]"
c'2 c'
```

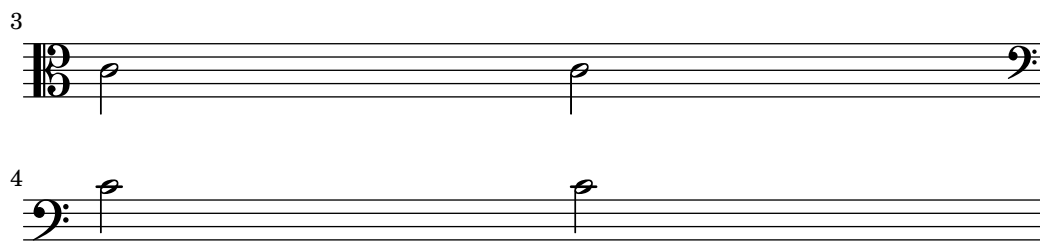


Le altezze vengono mostrate come se l'argomento numerico fosse inserito senza parentesi.

Se c'è un cambio di chiave quando si interrompe la linea, il simbolo della nuova chiave viene ripetuto alla fine della linea precedente, come chiave di *avviso*, e all'inizio di quella successiva. Tale chiave di *precauzione* può essere soppressa.

```
\clef treble { c'2 c' } \break
\clef bass { c'2 c' } \break
\clef alto
\set Staff.explicitClefVisibility = #end-of-line-invisible
{ c'2 c' } \break
\unset Staff.explicitClefVisibility
\clef bass { c'2 c' } \break
```





Una chiave che è già stata visualizzata non viene ristampata se viene ripetuto lo stesso comando `\clef` e verrà quindi ignorata. Si può cambiare tale comportamento predefinito col comando `\set Staff.forceClef = ##t`.

```
\clef treble
c'1
\clef treble
c'1
\set Staff.forceClef = ##t
c'1
\clef treble
c'1
```



When there is a manual clef change, the glyph of the changed clef will be smaller than normal. This behaviour can be overridden.

```
\clef "treble"
c'1
\clef "bass"
c'1
\clef "treble"
c'1
\override Staff.Clef.full-size-change = ##t
\clef "bass"
c'1
\clef "treble"
c'1
\revert Staff.Clef.full-size-change
\clef "bass"
c'1
\clef "treble"
c'1
```



## Frammenti di codice selezionati

### *Modifiche manuali della proprietà della chiave*

Cambiando il glifo della chiave, la sua posizione o l'ottavazione non cambia la posizione delle note successive nel rigo. Per far sì che le armature di chiave si trovino sulle linee del rigo corrette, bisogna specificare anche `middleCPosition`, con valori positivi o negativi che spostano il Do centrale rispettivamente su o giù in senso relativo alla linea centrale del rigo.

Per esempio, `\clef "treble_8"` equivale a impostare `clefGlyph`, `clefPosition` (che regola la posizione verticale della chiave), `middleCPosition` e `clefTransposition`. Viene stampata una chiave quando cambia una di queste proprietà, eccetto `middleCPosition`.

Gli esempi seguenti mostrano le possibilità date dall'impostazione manuale di tali proprietà. Sulla prima linea le modifiche manuali preservano il posizionamento relativo standard di chiavi e note, mentre sulla seconda linea non lo fanno.

```
{
% The default treble clef
\key f \major
c'1
% The standard bass clef
\set Staff.clefGlyph = #"clefs.F"
\set Staff.clefPosition = #2
\set Staff.middleCPosition = #6
\set Staff.middleCClefPosition = #6
\key g \major
c'1
% The baritone clef
\set Staff.clefGlyph = #"clefs.C"
\set Staff.clefPosition = #4
\set Staff.middleCPosition = #4
\set Staff.middleCClefPosition = #4
\key f \major
c'1
% The standard choral tenor clef
\set Staff.clefGlyph = #"clefs.G"
\set Staff.clefPosition = #-2
\set Staff.clefTransposition = #-7
\set Staff.middleCPosition = #1
\set Staff.middleCClefPosition = #1
\key f \major
c'1
% A non-standard clef
\set Staff.clefPosition = #0
\set Staff.clefTransposition = #0
\set Staff.middleCPosition = #-4
\set Staff.middleCClefPosition = #-4
\key g \major
c'1 \break

% The following clef changes do not preserve
% the normal relationship between notes, key signatures
% and clefs:

\set Staff.clefGlyph = #"clefs.F"
\set Staff.clefPosition = #2
c'1
\set Staff.clefGlyph = #"clefs.G"
c'1
\set Staff.clefGlyph = #"clefs.C"
c'1
```

```

\set Staff.clefTransposition = #7
c'1
\set Staff.clefTransposition = #0
\set Staff.clefPosition = #0
c'1

% Return to the normal clef:

\set Staff.middleCPosition = #0
c'1
}

```



## Vedi anche

Guida alla notazione: [Mensural clefs], pagina 436, [Gregorian clefs], pagina 443, [Default tablatures], pagina 341, [Custom tablatures], pagina 356, [Formatting cue notes], pagina [undefined].

File installati: `scm/parser-clef.scm`.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Clef\_engraver” in *Guida al Funzionamento Interno*, Sezione “Clef” in *Guida al Funzionamento Interno*, Sezione “ClefModifier” in *Guida al Funzionamento Interno*, Sezione “clef-interface” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

I numeri di ottavazione assegnati alle chiavi sono trattati come oggetti grafici separati. Quindi qualsiasi `\override` all’oggetto *Clef* dovrà essere applicato, con un altro `\override`, all’oggetto *ClefModifier*.

```

\new Staff \with {
  \override Clef.color = #blue
  \override ClefModifier.color = #red
}

\clef "treble_8" c'4

```



## Armatura di chiave

**Nota:** I nuovi utenti sono talvolta confusi dalla gestione delle alterazioni e delle armature di chiave. In LilyPond i nomi delle note costituiscono l'input grezzo; le armature e le chiavi determinano come questo venga mostrato. Una nota non alterata come `c` significa 'Do naturale', indipendentemente dall'armatura o dalla chiave. Per maggiori informazioni si veda Sezione "Altezze e armature di chiave" in *Manuale di Apprendimento*.

L'armatura di chiave indica la tonalità di un brano. È costituita da un insieme di alterazioni (bemolle o diesis) all'inizio del rigo. L'armatura di chiave può essere modificata:

```
\key altezza modo
```

`modo` deve essere `\major` o `\minor` per ottenere rispettivamente un'armatura di *altezza*-maggiore o *altezza*-minore. È anche possibile usare i nomi tradizionali dei modi, chiamati anche *modi ecclesiastici*: `\ionian`, `\dorian`, `\phrygian`, `\lydian`, `\mixolydian`, `\aeolian` e `\locrian`.

```
\relative {
  \key g \major
  fis''1
  f
  fis
}
```



Si possono definire ulteriori modi elencando le alterazioni per ogni grado della scala quando il modo inizia col Do.

```
freygish = #`((0 . ,NATURAL) (1 . ,FLAT) (2 . ,NATURAL)
              (3 . ,NATURAL) (4 . ,NATURAL) (5 . ,FLAT) (6 . ,FLAT))
```

```
\relative {
  \key c \freygish c'4 des e f
  \bar "||" \key d \freygish d es fis g
}
```



Le alterazioni dell'armatura di chiave possono essere collocate in posizioni diverse da quelle tradizionali o anche in più di un'ottava, usando le proprietà `flat-positions` e `sharp-positions` di `KeySignature`. I valori di queste proprietà specificano l'estensione delle posizioni del rigo in cui potranno comparire le alterazioni. Se viene specificata una sola posizione, le alterazioni vengono collocate entro l'ottava che finisce in quella posizione del rigo.

```
\override Staff.KeySignature.flat-positions = #'((-5 . 5))
\override Staff.KeyCancellation.flat-positions = #'((-5 . 5))
\clef bass \key es \major es g bes d'
\clef treble \bar "||" \key es \major es' g' bes' d''
```

```
\override Staff.KeySignature.sharp-positions = #'(2)
\bar "||" \key b \major b' fis' b'2
```



## Frammenti di codice selezionati

*Impedire l'inserimento dei segni di bequadro quando cambia l'armatura di chiave*

Quando l'armatura di chiave cambia, vengono inseriti automaticamente i segni di bequadro per annullare le alterazioni di precedenti armature. Si può evitare questo comportamento impostando su f (falso) la proprietà `printKeyCancellation` nel contesto `Staff`.

```
\relative c' {
  \key d \major
  a4 b cis d
  \key g \minor
  a4 bes c d
  \set Staff.printKeyCancellation = ##f
  \key d \major
  a4 b cis d
  \key g \minor
  a4 bes c d
}
```



*Armature di chiave non tradizionali*

Il comando `\key` comunemente usato imposta la proprietà `keyAlterations`, che fa parte del contesto `Staff`.

Per creare armature di chiave non standard, tale proprietà va impostata esplicitamente. Il formato di questo comando è una lista:

`\set Staff.keyAlterations = #`(((ottava . grado) . alterazione) ((ottava . grado) . alterazione) ...)` dove, per ogni elemento della lista, `ottava` indica l'ottava (0 è l'ottava dal Do centrale al Si precedente), `grado` indica la nota all'interno dell'ottava (0 significa Do e 6 significa Si) e `alterazione` può essere `,SHARP`, `,FLAT`, `,DOUBLE-SHARP` etc. (Si noti la virgola iniziale.)

Altrimenti, usando, per ogni elemento della lista, il formato breve `(grado . alterazione)`, ciò indica che la stessa alterazione deve essere presente in tutte le ottave.

Ecco un esempio di una possibile armatura per generare una scala a tono intero:

```
\relative {
  \set Staff.keyAlterations = #`((6 . ,FLAT)
                                (5 . ,FLAT)
                                (3 . ,SHARP))

  c'4 d e fis
  aes4 bes c2
}
```





## Vedi anche

Glossario musicale: Sezione “church mode” in *Glossario Musicale*, Sezione “scordatura” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Altezze e armature di chiave” in *Manuale di Apprendimento*.

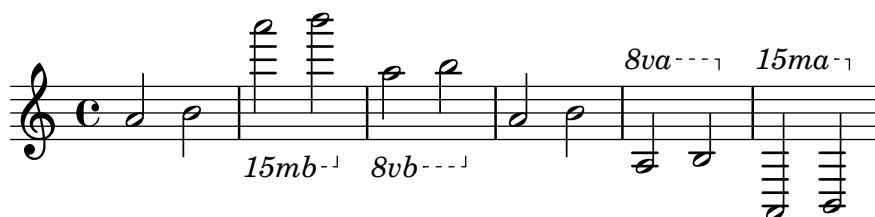
Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “KeyChangeEvent” in *Guida al Funzionamento Interno*, Sezione “Key\_engraver” in *Guida al Funzionamento Interno*, Sezione “Key\_performer” in *Guida al Funzionamento Interno*, Sezione “KeyCancellation” in *Guida al Funzionamento Interno*, Sezione “KeySignature” in *Guida al Funzionamento Interno*, Sezione “key-signature-interface” in *Guida al Funzionamento Interno*.

## Segni di ottavazione

I *segni di ottavazione* introducono un’ulteriore trasposizione di ottava nel rigo:

```
\relative a' {
  a2 b
  \ottava #-2
  a2 b
  \ottava #-1
  a2 b
  \ottava #0
  a2 b
  \ottava #1
  a2 b
  \ottava #2
  a2 b
}
```



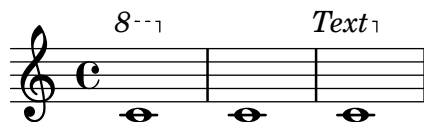
## Frammenti di codice selezionati

### Testo dell’ottava

Internamente, `\ottava` imposta le proprietà `ottavation` (ad esempio, su `8va` o `8vb`) e `middleCPosition`. Per sovrascrivere il testo della parentesi, occorre specificare `ottavation` dopo il comando `\ottava`.

```
{
  \ottava #1
  \set Staff.ottavation = #"8"
  c'1
  \ottava #0
  c'1
  \ottava #1
```

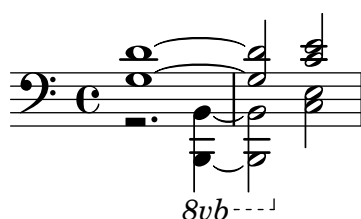
```
\set Staff.ottavation = #"Text"
c' '1
}
```



### *Aggiungere un segno di ottava a una sola voce*

Se il rigo ha più di una voce, l'ottavazione in una voce trasporrà la posizione delle note in tutte le voci per la durata della parentesi dell'ottava. Se si intende applicare l'ottavazione a una sola voce, si possono impostare esplicitamente `middleCPosition` e la parentesi di ottava. In questo frammento, la chiave di basso ha di norma il `MiddleCPosition` impostato su 6, ovvero sei posizioni sopra la linea centrale, dunque nella porzione con l'ottava il `MiddleCPosition` è più alto di sette posizioni (un'ottava).

```
{
  \clef bass
  << { <g d'>1~ q2 <c' e'> }
  \\
  {
    r2.
    \set Staff.ottavation = #"8vb"
    \once \override Staff.OttavaBracket.direction = #DOWN
    \set Voice.middleCPosition = #(+ 6 7)
    <b,,, b,,,>4 ~ |
    q2
    \unset Staff.ottavation
    \unset Voice.middleCPosition
    <c e>2
  }
  >>
}
```



### *Modificare l'inclinazione dell'estensore dell'ottava*

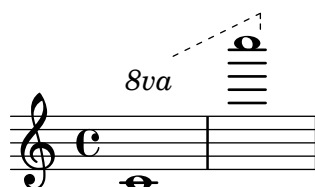
È possibile cambiare l'inclinazione dell'estensore dell'ottava.

```
\relative c' {
  \override Staff.OttavaBracket.stencil = #ly:line-spanner::print
  \override Staff.OttavaBracket.bound-details =
    #`((left . ((Y . 0) ; Change the integer here
      (attach-dir . ,LEFT)
      (padding . 0)
      (stencil-align-dir-y . ,CENTER)))
    (right . ((Y . 5) ; Change the integer here
      (padding . 0))
  }
```

```

      (attach-dir . ,RIGHT)
      (text . ,(make-draw-dashed-line-markup (cons 0 -1.2))))))
\override Staff.OttavaBracket.left-bound-info =
  #ly:line-spanner::calc-left-bound-info-and-text
\override Staff.OttavaBracket.right-bound-info =
  #ly:line-spanner::calc-right-bound-info
\ottava #1
c1
c'''1
}

```



## Vedi anche

Glossario musicale: Sezione “ottavazione” in *Glossario Musicale*.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Ottava\_spanner\_engraver” in *Guida al Funzionamento Interno*, Sezione “OttavaBracket” in *Guida al Funzionamento Interno*, Sezione “ottava-bracket-interface” in *Guida al Funzionamento Interno*.

## Trasporto strumentale

Quando si scrivono partiture che comprendono strumenti traspositori, alcune parti possono essere scritte a un'altezza diversa dall'*intonazione reale*. In questi casi, è necessario specificare la chiave dello *strumento traspositore*, altrimenti l'output MIDI e le citazioni in altre parti produrranno altezze errate. Per maggiori informazioni sulle citazioni, si veda `<undefined>` [Quoting other voices], pagina `<undefined>`.

`\transposition altezza`

L'altezza da usare per `\transposition` deve corrispondere al suono effettivamente prodotto quando un `c'` scritto sul rigo viene suonato dallo strumento traspositore. Tale altezza viene inserita in modalità assoluta; dunque, uno strumento che produce un suono reale un tono sopra la notazione deve usare `\transposition d'`. `\transposition` va usato *soltanto* se le altezze *non* sono scritte in intonazione reale.

Ecco un frammento per violino e clarinetto in Si bemolle, le cui parti sono inserite usando le note e l'armatura di chiave che appaiono nei rispettivi rigi sulla partitura del direttore. I due strumenti suonano all'unisono.

```

\new GrandStaff <<
  \new Staff = "violin" {
    \relative c'' {
      \set Staff.instrumentName = #"Vln"
      \set Staff.midiInstrument = #"violin"
      % not strictly necessary, but a good reminder
      \transposition c'

      \key c \major
      g4( c8) r c r c4
    }
  }

```

```

    }
  }
  \new Staff = "clarinet" {
    \relative c'' {
      \set Staff.instrumentName = \markup { Cl (B\flat) }
      \set Staff.midiInstrument = #"clarinet"
      \transposition bes

      \key d \major
      a4( d8) r d r d4
    }
  }
  >>

```



`\transposition` può essere modificato nel corso di un brano. Ad esempio, un clarinettista potrebbe essere costretto a passare da un clarinetto in La a uno in Si bemolle.

```

flute = \relative c'' {
  \key f \major
  \cueDuring #"clarinet" #DOWN {
    R1 _\markup\tiny "clarinet"
    c4 f e d
    R1 _\markup\tiny "clarinet"
  }
}
clarinet = \relative c'' {
  \key aes \major
  \transposition a
  aes4 bes c des
  R1^\markup { muta in B\flat }
  \key g \major
  \transposition bes
  d2 g,
}
\addQuote "clarinet" \clarinet
<<
  \new Staff \with { instrumentName = #"Flute" }
  \flute
  \new Staff \with { instrumentName = #"Cl (A)" }
  \clarinet
>>

```



## Vedi anche

Glossario musicale: Sezione “intonazione reale” in *Glossario Musicale*, Sezione “strumento traspositore” in *Glossario Musicale*.

Guida alla notazione: [\[Quoting other voices\]](#), pagina [\[Transpose\]](#), pagina [\[Transpose\]](#).

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

## Alterazioni automatiche

Esistono diverse convenzioni sul modo di scrivere le alterazioni. LilyPond ha una funzione per specificare lo stile di gestione delle alterazioni adottato. Questa funzione viene richiamata nel modo seguente:

```
\new Staff <<
  \accidentalStyle voice
  { ... }
>>
```

La gestione delle alterazioni si applica di norma all'attuale **Staff** (con l'eccezione degli stili **piano** e **piano-cautionary**, che sono spiegati dopo). Questa funzione accetta un secondo argomento opzionale che determina in quale ambito debba essere cambiato lo stile. Ad esempio, per usare lo stesso stile in tutti i righe dell'attuale **StaffGroup**, si usa:

```
\accidentalStyle StaffGroup.voice
```

Sono supportati i seguenti modi di gestire le alterazioni. Il seguente esempio mostra tutti gli stili:

```
musicA = {
  <<
    \relative {
      cis''8 fis, bes4 <a cis>8 f bis4 |
      cis2. <c, g'>4 |
    }
    \\\
    \relative {
      ais'2 cis, |
      fis8 b a4 cis2 |
    }
  >>
}

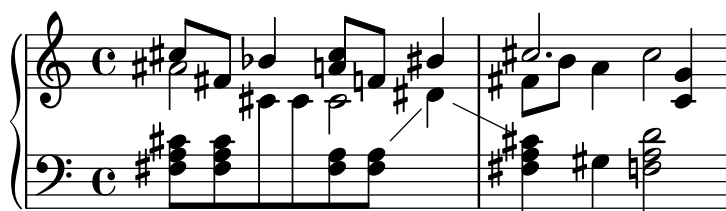
musicB = {
  \clef bass
  \new Voice {
    \voiceTwo \relative {
      <fis a cis>8[ <fis a cis>
      \change Staff = up
      cis' cis
      \change Staff = down
```

```

    <fis, a> <fis a>]
    \showStaffSwitch
    \change Staff = up
    dis'4 |
    \change Staff = down
    <fis, a cis>4 gis <f a d>2 |
  }
}
}

\new PianoStaff {
  <<
    \context Staff = "up" {
      \accidentalStyle default
      \musicA
    }
    \context Staff = "down" {
      \accidentalStyle default
      \musicB
    }
  >>
}

```



Si noti che le ultime linee di questo esempio possono essere sostituite dal seguente frammento, se si vuole usare lo stesso stile in entrambi i righe.

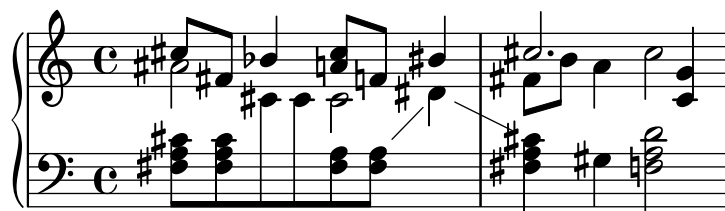
```

\new PianoStaff {
  <<
    \context Staff = "up" {
      %% change the next line as desired:
      \accidentalStyle Score.default
      \musicA
    }
    \context Staff = "down" {
      \musicB
    }
  >>
}

default

```

Questo è il comportamento predefinito del compositore tipografico. Corrisponde alla pratica comunemente impiegata dal diciottesimo secolo: le alterazioni vengono ricordate fino alla fine della misura in cui si trovano, limitatamente all'ottava di appartenenza. Quindi, nell'esempio seguente non compare alcun segno di bequadro prima del **b** nella seconda misura o prima dell'ultimo **c**:

**voice**

Normalmente le alterazioni mantengono la propria validità a livello di **Staff**. Tuttavia in questo stile le alterazioni vengono gestite individualmente per ogni voce. Al di fuori di quest'aspetto, lo stile è analogo a **default**.

Di conseguenza, le alterazioni relative a una voce non vengono cancellate nelle altre voci. Un risultato spesso non desiderabile: nell'esempio seguente è difficile capire se il secondo **a** sia naturale o diesis. L'opzione **voice** deve essere quindi usata solo se ogni voce è destinata a un esecutore diverso. Se la partitura deve essere letta da un unico musicista (come nel caso della partitura del direttore, o di uno spartito per pianoforte), allora è preferibile usare **modern** o **modern-cautionary**.

**modern**

Questa regola corrisponde alla pratica comune del ventesimo secolo. Omette i segni di bequadro supplementari che in passato erano di norma anteposti al diesis che segue un doppio diesis o a un bemolle che segue un doppio bemolle. La regola **modern** presenta le stesse alterazioni di **default**, con due aggiunte che servono a evitare ambiguità: i segni di annullamento delle alterazioni temporanee sono anteposti alle note sulla stessa ottava della misura successiva e alle note in ottave diverse nella stessa misura. In questo esempio, dunque, i bequadri del **b** e del **c** nella seconda misura del rigo superiore:

**modern-cautionary**

Questa regola è simile a **modern**, ma le alterazioni 'supplementari' sono segnate come alterazioni di precauzione (con parentesi). La loro dimensione può essere cambiata attraverso la proprietà **font-size** di **AccidentalCautionary**.



**modern-voice**

Questa regola viene usata per le alterazioni su più voci destinate sia agli esecutori che suonano una singola voce sia a quelli che suonano tutte le voci. Le alterazioni sono mostrate su tutte le voci, ma *sono annullate* su ogni voce dello stesso rigo (Staff). Quindi, l'alterazione dell' *a* nell'ultima misura viene annullata perché l'annullamento precedente si trovava in una voce diversa, mentre quella del *d* nel rigo inferiore viene annullata a causa dell'alterazione in un'altra voce della misura precedente:

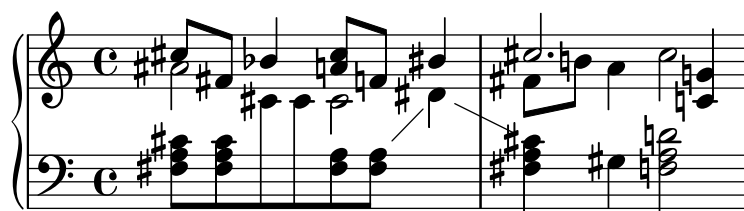
**modern-voice-cautionary**

Questa regola è analoga a **modern-voice**, ma con le alterazioni supplementari (quelle non mostrate da **voice**) segnate come alterazioni di precauzione. Tutte le alterazioni mostrate da **default** *sono* mostrate con questa regola, ma alcune di esse sono indicate come alterazioni di precauzione.

**piano**

Questa regola riflette la pratica del ventesimo secolo per la notazione per pianoforte. Il suo comportamento è molto simile allo stile **modern**, ma in questo caso le alterazioni vengono annullate in tutti i righi che si trovano nello stesso **GrandStaff** o **PianoStaff**, dunque tutte gli annullamenti delle note finali.

È lo stile predefinito per gli attuali **GrandStaff** e **PianoStaff**.

**piano-cautionary**

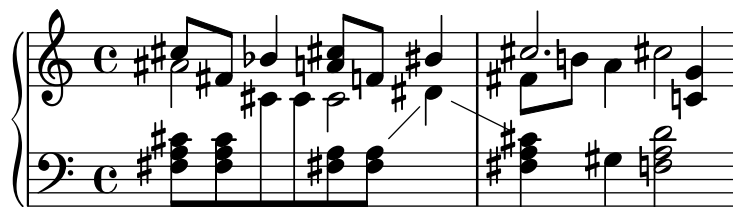
È uguale a **piano** ma con le alterazioni supplementari mostrate come alterazioni di precauzione.





**neo-modern**

Questa regola si riferisce a una pratica tipica della musica contemporanea: le alterazioni sono mostrate come in **modern**, ma vengono ripetute se la stessa nota appare in seguito nella stessa misura – a meno che la seconda occorrenza non segua direttamente la prima.

**neo-modern-cautionary**

Questa regola è simile a **neo-modern**, ma le alterazioni ‘supplementari’ sono mostrate come alterazioni di precauzione (con parentesi). La loro dimensione può essere modificata attraverso la proprietà `font-size` di `AccidentalCautionary`.

**neo-modern-voice**

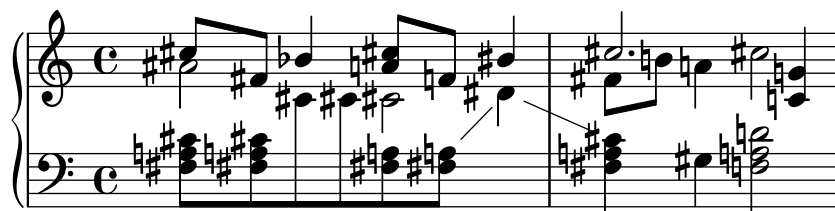
Questa regola viene usata per le alterazioni su più di una voce che devono essere lette sia da musicisti che suonano una singola voce sia da musicisti che suonano tutte le voci. Le alterazioni per ogni voce sono mostrate come nello stile **neo-modern**, ma vengono annullate attraverso le voci nello stesso rigo (`Staff`).

**neo-modern-voice-cautionary**

Questa regola è simile a **neo-modern-voice**, ma le alterazioni supplementari sono indicate come alterazioni di precauzione.

**dodecaphonic**

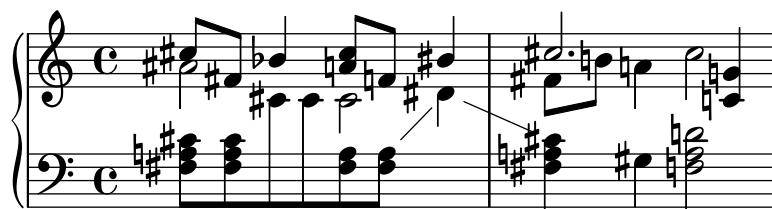
Questa regola riflette una regola introdotta dai compositori all’inizio del ventesimo secolo nel tentativo di abolire la gerarchia tra suoni naturali e non naturali. Con questo stile, *ogni* nota presenta un segno di alterazione, anche i suoni naturali.

**dodecaphonic-no-repeat**

Come nello stile delle alterazioni dodecafonico *ogni* nota ha un segno di alterazione, ma le alterazioni sono soppresse per tutte le altezze ripetute immediatamente nello stesso rigo.

**dodecaphonic-first**

In modo analogo allo stile delle alterazioni dodecafonico *ogni* altezza ha un segno di alterazione, ma solo la prima volta che si incontra in una misura. Le alterazioni vengono ricordate solo per l'ottava corrente ma in tutte le voci.

**teaching**

Questa regola è pensata per gli studenti: permette di generare facilmente degli spartiti di scale con le alterazioni di precauzione inserite in modo automatico. Alle alterazioni, indicate come nello stile **modern**, vengono aggiunte ulteriori segni di precauzione per tutti i diesis e bemolle specificati dall'armatura di chiave, fuorché nel caso di ripetizioni immediatamente successive di una stessa nota.

**no-reset**

È identico a **default**, ma le alterazioni mantengono la propria validità 'per sempre', non solo all'interno della singola misura:



**forget**

È il contrario di **no-reset**: le alterazioni non vengono ricordate affatto – pertanto, tutte le alterazioni si riferiscono all’armatura di chiave, indipendentemente dal materiale musicale precedente.

**Vedi anche**

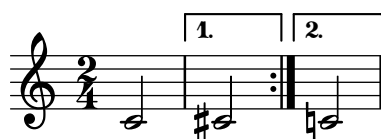
Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Accidental” in *Guida al Funzionamento Interno*, Sezione “Accidental\_engraver” in *Guida al Funzionamento Interno*, Sezione “GrandStaff” in *Guida al Funzionamento Interno*, Sezione “PianoStaff” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*, Sezione “AccidentalSuggestion” in *Guida al Funzionamento Interno*, Sezione “AccidentalPlacement” in *Guida al Funzionamento Interno*, Sezione “accidental-suggestion-interface” in *Guida al Funzionamento Interno*.

**Problemi noti e avvertimenti**

Le note simultanee non vengono considerate nell’individuazione automatica delle alterazioni; vengono prese come riferimento solo le note precedenti e l’armatura di chiave. Se la stessa nota occorre simultaneamente con alterazioni diverse, può essere necessario forzare le alterazioni con ! o ?: ‘<f! fis!>’.

L’annullamento di precauzione delle alterazioni avviene in relazione alla misura precedente. Tuttavia, nel blocco `\alternative` che segue una sezione `\repeat volta N`, è auspicabile che l’annullamento sia calcolato in base alla precedente misura *eseguita*, non alla precedente misura *stampata*. Nell’esempio seguente il Do naturale della seconda volta non richiede il segno di bequadro:



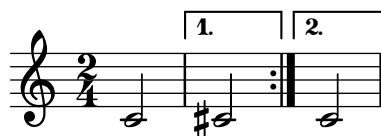
Si può usare il seguente espediente: si definisce una funzione che imposti localmente lo stile delle alterazioni su **forget**:

```
forget = #(define-music-function (music) (ly:music?) #{
  \accidentalStyle forget
  #music
  \accidentalStyle modern
#})
{
  \accidentalStyle modern
  \time 2/4
  \repeat volta 2 {
    c'2
  }
}
```

```

\alternative {
  cis'
  \forget c'
}

```



## Ambitus

Il termine *ambitus* (pl. *ambitus*) indica l'ambito di altezze di una determinata voce all'interno di una composizione musicale. Può indicare anche l'estensione di uno strumento musicale, ovvero l'intera gamma di suoni che può produrre. L'*ambitus* viene usato nelle parti vocali in modo che gli esecutori possano capire facilmente se siano adeguate alle loro possibilità.

L'*ambitus* viene indicato all'inizio del brano, prima della chiave iniziale. L'intervallo è individuato graficamente da due teste di nota che rappresentano l'altezza più bassa e più alta. Le alterazioni sono mostrate solo se non fanno parte dell'armatura di chiave.

```

\layout {
  \context {
    \Voice
    \consists "Ambitus_engraver"
  }
}

\relative {
  aes' c e2
  cis,1
}

```



## Frammenti di codice selezionati

### *Un ambitus per voce*

L'*ambitus* può essere specificato per voce. In tal caso occorre spostarlo manualmente per evitare collisioni.

```

\new Staff <<
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c'' {
    \override Ambitus.X-offset = #2.0
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \with {
    \consists "Ambitus_engraver"
  }

```

```

    } \relative c' {
      \voiceTwo
      es4 f g as
      b1
    }
  >>

```



### *Ambitus su più voci*

Se si aggiunge l'incisore `Ambitus_engraver` al contesto `Staff` viene creato un solo ambitus per il rigo, anche nel caso di rigi che hanno più voci.

```

\new Staff \with {
  \consists "Ambitus_engraver"
}
<<
  \new Voice \relative c'' {
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \relative c' {
    \voiceTwo
    es4 f g as
    b1
  }
}
>>

```



### *Modifica dell'intervallo dell'ambitus*

È possibile cambiare le impostazioni predefinite dell'intervallo tra le teste di nota dell'ambitus e la linea che le collega.

```

\layout {
  \context {
    \Voice
    \consists "Ambitus_engraver"
  }
}

\new Staff {
  \time 2/4
  % Default setting
  c'4 g''
}

```

```

\new Staff {
  \time 2/4
  \override AmbitusLine.gap = #0
  c'4 g''
}

\new Staff {
  \time 2/4
  \override AmbitusLine.gap = #1
  c'4 g''
}

\new Staff {
  \time 2/4
  \override AmbitusLine.gap = #1.5
  c'4 g''
}

```



## Vedi anche

Glossario musicale: Sezione “ambitus” in *Glossario Musicale*.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Ambitus\_engraver” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*, Sezione “Ambitus” in *Guida al Funzionamento Interno*, Sezione “AmbitusAccidental” in *Guida al Funzionamento Interno*, Sezione “AmbitusLine” in *Guida al Funzionamento Interno*, Sezione “AmbitusNoteHead” in *Guida al Funzionamento Interno*, Sezione “ambitus-interface” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Le collisioni non vengono gestite in presenza di un ambitus multiplo su più di una voce.

### 1.1.4 Teste di nota

Questa sezione suggerisce i modi in cui modificare la testa di una nota.

## Teste di nota speciali

L'aspetto delle teste delle note può essere modificato:

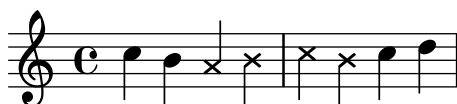
```
\relative c'' {
  c4 b
  \override NoteHead.style = #'cross
  c4 b
  \revert NoteHead.style
  a b
  \override NoteHead.style = #'harmonic
  a b
  \revert NoteHead.style
  c4 d e f
}
```



L'elenco di tutti gli stili per le teste di nota è in [\[Note head styles\]](#), pagina [\[undefined\]](#).

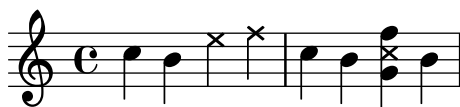
Lo stile barrato (**cross**) viene usato per rappresentare varie intenzioni musicali. I seguenti comandi generici predefiniti modificano la testa della nota nei contesti del rigo e dell'intavolatura e possono essere usati per rappresentare qualsiasi significato musicale:

```
\relative {
  c''4 b
  \xNotesOn
  a b c4 b
  \xNotesOff
  c4 d
}
```



Questo comando può essere usato all'interno e all'esterno degli accordi per generare teste barrate sia nel contesto del rigo che in quello dell'intavolatura:

```
\relative {
  c''4 b
  \xNote { e f }
  c b < g \xNote c f > b
}
```



Potete utilizzare, al posto di `\xNote`, `\xNotesOn` e `\xNotesOff`, i comandi `\deadNote`, `\deadNotesOn` e `\deadNotesOff`. Il termine *dead note* è di uso comune tra i chitarristi.

Esiste anche una scorciatoia simile per le forme a diamante:

```
\relative c'' {
```

```
<c f\harmonic>2 <d a'\harmonic>4 <c g'\harmonic> f\harmonic
}
```



## Comandi predefiniti

`\harmonic`, `\xNotesOn`, `\xNotesOff`, `\xNote`.

## Vedi anche

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida alla notazione: `<undefined>` [Note head styles], pagina `<undefined>`, `<undefined>` [Chord-ed notes], pagina `<undefined>`, [Indicating harmonics and dampened notes], pagina 384.

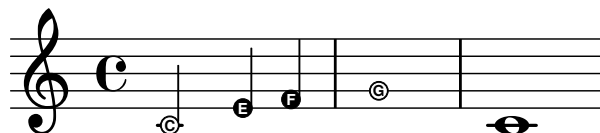
Guida al funzionamento interno: Sezione “note-event” in *Guida al Funzionamento Interno*, Sezione “Note\_heads\_engraver” in *Guida al Funzionamento Interno*, Sezione “Ledger\_line\_engraver” in *Guida al Funzionamento Interno*, Sezione “NoteHead” in *Guida al Funzionamento Interno*, Sezione “LedgerLineSpanner” in *Guida al Funzionamento Interno*, Sezione “note-head-interface” in *Guida al Funzionamento Interno*, Sezione “ledger-line-spanner-interface” in *Guida al Funzionamento Interno*.

## Testa di nota con nome della nota

La nota ‘easy play’ inserisce il nome della nota dentro la testa. Viene usata nella musica per principianti. Per rendere le lettere leggibili, occorrerebbe usare un carattere più grande. A questo proposito si veda `<undefined>` [Setting the staff size], pagina `<undefined>`.

```

#(set-global-staff-size 26)
\relative c' {
  \easyHeadsOn
  c2 e4 f
  g1
  \easyHeadsOff
  c,1
}
```



## Comandi predefiniti

`\easyHeadsOn`, `\easyHeadsOff`.

## Frammenti di codice selezionati

*Numeri dentro le teste di nota*

Le teste di nota con nome della nota usano la proprietà `note-names` dell’oggetto `NoteHead` per determinare cosa appaia all’interno della testa. È possibile sovrascrivere questa proprietà e mostrare numeri corrispondenti ai gradi della scala.

Si può creare un semplice incisore che faccia questo per ogni oggetto testa di nota che incontra.

```
#(define Ez_numbers_engraver
```



```

(make-engraver
  (acknowledgers
    ((note-head-interface engraver grob source-engraver)
      (let* ((context (ly:translator-context engraver))
              (tonic-pitch (ly:context-property context 'tonic))
              (tonic-name (ly:pitch-notename tonic-pitch))
              (grob-pitch
                (ly:event-property (event-cause grob) 'pitch))
              (grob-name (ly:pitch-notename grob-pitch))
              (delta (modulo (- grob-name tonic-name) 7))
              (note-names
                (make-vector 7 (number->string (1+ delta)))))
        (ly:grob-set-property! grob 'note-names note-names))))))

#(set-global-staff-size 26)

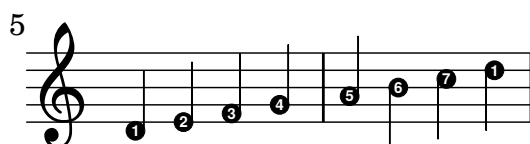
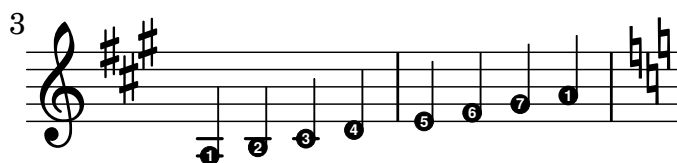
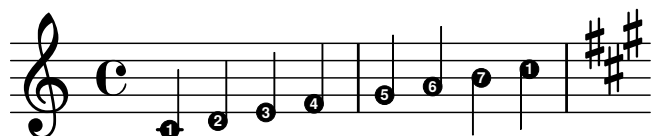
\layout {
  ragged-right = ##t
  \context {
    \Voice
    \consists \Ez_numbers_engraver
  }
}

\relative c' {
  \easyHeadsOn
  c4 d e f
  g4 a b c \break

  \key a \major
  a,4 b cis d
  e4 fis gis a \break

  \key d \dorian
  d,4 e f g
  a4 b c d
}

```



## Vedi anche

Guida alla notazione: `<undefined>` [Setting the staff size], pagina `<undefined>`.

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “note-event” in *Guida al Funzionamento Interno*, Sezione “Note\_heads\_engraver” in *Guida al Funzionamento Interno*, Sezione “NoteHead” in *Guida al Funzionamento Interno*, Sezione “note-head-interface” in *Guida al Funzionamento Interno*.

## Teste di nota a forma variabile

In alcune notazioni, la forma della testa della nota corrisponde alla funzione armonica di una nota nella scala. Questa notazione era comune nei canzonieri americani del diciannovesimo secolo. Gli stili possibili sono Sacred Harp, Southern Harmony, Funk (Harmonica Sacra), Walker e Aiken (Christian Harmony):

```
\relative c'' {
  \aikenHeads
  c, d e f g2 a b1 c \break
  \sacredHarpHeads
  c,4 d e f g2 a b1 c \break
  \southernHarmonyHeads
  c,4 d e f g2 a b1 c \break
  \funkHeads
  c,4 d e f g2 a b1 c \break
  \walkerHeads
  c,4 d e f g2 a b1 c \break
}
```



Le forme variano in base al grado della scala; la scala è determinata dal comando `\key`. Se si scrive in tonalità minore, il grado della scala può essere determinato in base alla relativa maggiore:

```
\relative c'' {
```

```

\key a \minor
\aikenHeads
a b c d e2 f g1 a \break
\aikenHeadsMinor
a,4 b c d e2 f g1 a \break
\sacredHarpHeadsMinor
a,2 b c d \break
\southernHarmonyHeadsMinor
a2 b c d \break
\funkHeadsMinor
a2 b c d \break
\walkerHeadsMinor
a2 b c d \break
}

```



## Comandi predefiniti

\aikenHeads, \aikenHeadsMinor, \funkHeads, \funkHeadsMinor, \sacredHarpHeads,  
\sacredHarpHeadsMinor, \southernHarmonyHeads, \southernHarmonyHeadsMinor,  
\walkerHeads, \walkerHeadsMinor.

## Frammenti di codice selezionati

### *Applicazione degli stili delle teste di nota in base al grado della scala*

La proprietà `shapeNoteStyles` può essere usata per definire vari stili di teste di nota per ogni grado della scala (definita dall'armatura di chiave o dalla proprietà `tonic`). Questa proprietà richiede un insieme di simboli, che può essere puramente arbitrario (sono permesse espressioni geometriche come `triangle`, `cross` e `xcircle`) o basato sull'antica tradizione tipografica americana (sono consentiti anche alcuni nomi di nota latini).

Detto questo, per imitare gli antichi canzonieri americani, ci sono vari stili predefiniti disponibili attraverso dei comodi comandi come `\aikenHeads` o `\sacredHarpHeads`.

Questo esempio mostra modi diversi di ottenere teste di nota di varie forme e illustra la possibilità di trasporre una melodia senza perdere la corrispondenza tra le funzioni armoniche e gli stili delle teste.

```

fragment = {
  \key c \major
  c2 d
  e2 f
  g2 a
  b2 c
}

\new Staff {
  \transpose c d
  \relative c' {
    \set shapeNoteStyles = ##(do re mi fa
                          #f la ti)

    \fragment
  }

  \break

  \relative c' {
    \set shapeNoteStyles = ##(cross triangle fa #f
                          mensural xcircle diamond)

    \fragment
  }
}

```



La lista completa di tutti gli stili delle teste si trova in `\aikenHeads` [Note head styles], pagina 44.

## Vedi anche

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida alla notazione: `<undefined>` [Note head styles], pagina `<undefined>`.

Guida al funzionamento interno: Sezione “note-event” in *Guida al Funzionamento Interno*, Sezione “Note\_heads\_engraver” in *Guida al Funzionamento Interno*, Sezione “NoteHead” in *Guida al Funzionamento Interno*, Sezione “note-head-interface” in *Guida al Funzionamento Interno*.

## Improvvisazione

L'improvvisazione viene talvolta indicata con teste tagliate: l'esecutore può scegliere qualsiasi nota ma deve seguire il ritmo indicato. Si possono creare queste teste:

```
\new Voice \with {
  \consists "Pitch_squash_engraver"
} \relative {
  e''8 e g a a16( bes) a8 g
  \improvisationOn
  e8 ~
  2 ~ 8 f4 f8 ~
  2
  \improvisationOff
  a16( bes) a8 g e
}
```



## Comandi predefiniti

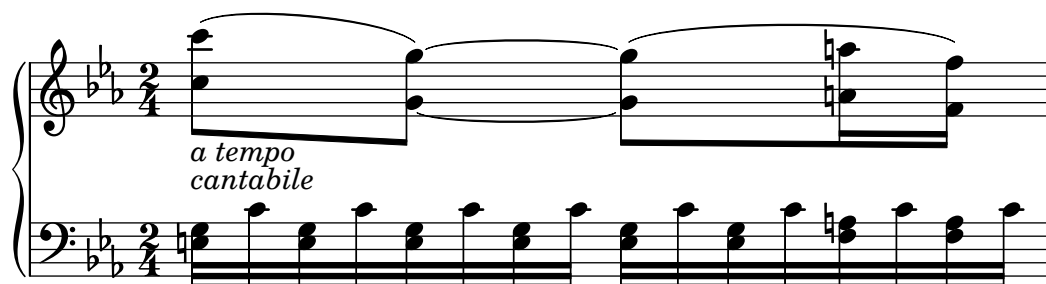
`\improvisationOn`, `\improvisationOff`.

## Vedi anche

Frammenti di codice: Sezione “Altezze” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Pitch\_squash\_engraver” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*, Sezione “RhythmicStaff” in *Guida al Funzionamento Interno*.

## 1.2 Ritmi



Questa sezione tratta i ritmi, le pause, le durate, la disposizione delle travature e le battute.

### 1.2.1 Inserimento delle durate

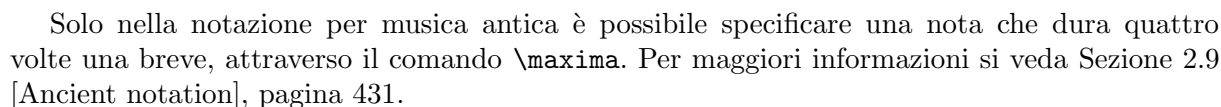
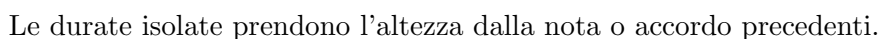
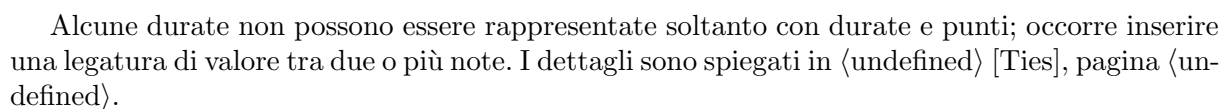
#### Durata

Le durate, indicate con numeri e punti, sono indicate con i valori corrispondenti. Per esempio, una nota di un quarto si indica con un 4 (dato che il suo valore è un  $1/4$ ), mentre una minima si indica col 2 (dato che il suo valore è  $1/2$ ). Per le note più lunghe di un intero bisogna usare i comandi `\longa` (due volte una breve) e `\breve`. La minor durata esprimibile per una nota indipendente è di 128; è possibile inserire anche valori inferiori, ma solo all'interno di travature.

```
\relative {
  \time 8/1
  c''\longa c\breve c1 c2
  c4 c8 c16 c32 c64 c128 c128
}
```

Ecco gli stessi valori con la disposizione automatica delle travature disabilitata.

```
\relative {
  \time 8/1
  \autoBeamOff
  c''\longa c\breve c1 c2
  c4 c8 c16 c32 c64 c128 c128
}
```


$$\backslash \text{relative} \{ a' \ a \ a_2 \ a \ a_4 \ a \ a_1 \ a \}$$
[illegible]
$$\backslash \text{relative} \{ a^4 b c^4. b^8 a^4. b^4.. c^8. \}$$


Le note possono essere distanziate in modo rigorosamente proporzionale alla loro durata. I dettagli relativi a questo argomento e alle impostazioni della notazione proporzionale si trovano in [\[Proportional notation\]](#), pagina [1](#).

Di norma i punti sono spostati in su per evitare le linee del rigo, fuorché all'interno di passaggi polifonici. I punti possono essere orientati manualmente verso l'alto o verso il basso; si veda Sezione 5.4.2 [Direction and placement], pagina 611.

## \autoBeamOn, \autoBeamOff, \dotsUp, \dotsDown, \dotsNeutral.

## Frammenti di codice selezionati

### *Note brevi alternative*

Le note brevi sono disponibili anche con due linee verticali su ciascun lato della testa invece di una sola e in stile barocco.

```
\relative c' {
  \time 4/2
  c\breve |
  \override Staff.NoteHead.style = #'altdefault
  b\breve
  \override Staff.NoteHead.style = #'baroque
  b\breve
  \revert Staff.NoteHead.style
  a\breve
}
```



### *Modifica del numero di punti di aumentazione per nota*

Il numero di punti di aumentazione su una singola nota può essere modificato in modo indipendente dai punti posizionati dopo la nota.

```
\relative c' {
  c4.. a16 r2 |
  \override Dots.dot-count = #4
  c4.. a16 r2 |
  \override Dots.dot-count = #0
  c4.. a16 r2 |
  \revert Dots.dot-count
  c4.. a16 r2 |
}
```



## Vedi anche

Glossario musicale: Sezione “breve” in *Glossario Musicale*, Sezione “longa” in *Glossario Musicale*, Sezione “maxima” in *Glossario Musicale*, Sezione “durata” in *Glossario Musicale*, Sezione “Nomi di durata delle note e delle pause” in *Glossario Musicale*.

Guida alla notazione: [Automatic beams](#), pagina [Ties](#), pagina [Stems](#), pagina [Writing rhythms](#), pagina [Writing rests](#), pagina [Vocal music](#), pagina [Ancient notation](#), pagina 431, [Proportional notation](#), pagina [Proportional notation](#).

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Dots” in *Guida al Funzionamento Interno*, Sezione “DotColumn” in *Guida al Funzionamento Interno*.



## Problemi noti e avvertimenti

Non c'è un limite massimo o minimo alla durata di una pausa, ma è il numero dei glifi ad essere limitato: si possono indicare pause da un centoventottesimo fino alla maxima (otto volte una semibreve).

## Gruppi irregolari

I gruppi irregolari sono costituiti da un'espressione musicale introdotta dal comando `\tuplet`, che moltiplica la velocità dell'espressione musicale per una frazione:

```
\tuplet frazione { musica }
```

Il numeratore della frazione apparirà sopra o sotto le note; eventualmente, con l'aggiunta opzionale di una parentesi quadra. Il gruppo irregolare più comune è la terzina, in cui 3 note hanno la durata di 2:

```
\relative {
  a'2 \tuplet 3/2 { b4 b b }
  c4 c \tuplet 3/2 { b4 a g }
}
```



In caso di lunghi passaggi di gruppi irregolari, dover scrivere un comando `\tuplet` per ogni gruppo è scomodo. È possibile specificare direttamente la durata di un gruppo irregolare prima della musica per far sì che i gruppi siano suddivisi automaticamente:

```
\relative {
  g'2 r8 \tuplet 3/2 8 { cis16 d e e f g g f e }
}
```



Le parentesi dei gruppi irregolari si possono posizionare manualmente sopra o sotto il rigo:

```
\relative {
  \tupletUp \tuplet 3/2 { c''8 d e }
  \tupletNeutral \tuplet 3/2 { c8 d e }
  \tupletDown \tuplet 3/2 { f,8 g a }
  \tupletNeutral \tuplet 3/2 { f8 g a }
}
```



È possibile annidare i gruppi irregolari:

```
\relative {
  \autoBeamOff
  c''4 \tuplet 5/4 { f8 e f \tuplet 3/2 { e[ f g] } } f4
}
```



La modifica di gruppi irregolari annidati che iniziano simultaneamente richiede l'uso di `\tweak`.

Per modificare la durata delle note senza che appaia la parentesi quadra del gruppo irregolare, si veda `\tweak` [Scaling durations], pagina `\tweak`.

## Comandi predefiniti

`\tupletUp`, `\tupletDown`, `\tupletNeutral`.

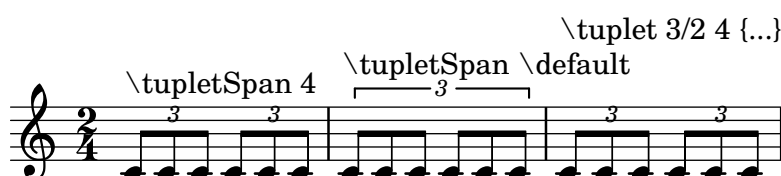
## Frammenti di codice selezionati

*Inserire vari gruppi irregolari usando una sola volta il comando `\tuplet`*

La proprietà `tupletSpannerDuration` imposta la durata di ognuno dei gruppi irregolari compresi tra parentesi dopo il comando `\tuplet`. In questo modo si possono inserire molti gruppi irregolari consecutivi all'interno di una singola espressione `\tuplet`, risparmiando così tempo e spazio.

Ci sono vari modi per impostare `tupletSpannerDuration`. Il comando `\tupletSpan` la imposta su una certa durata e poi la annulla quando invece di una durata viene specificato `\default`. Altrimenti si può usare un argomento opzionale con `\tuplet`.

```
\relative c' {
  \time 2/4
  \tupletSpan 4
  \tuplet 3/2 { c8^"\tupletSpan 4" c c c c c }
  \tupletSpan \default
  \tuplet 3/2 { c8^"\tupletSpan \default" c c c c c }
  \tuplet 3/2 4 { c8^"\tuplet 3/2 4 {...}" c c c c c }
}
```



*Modifica del numero del gruppo irregolare*

Di norma compare sulla parentesi del gruppo irregolare solo il numeratore del numero del gruppo irregolare, ovvero il numeratore dell'argomento del comando `\tuplet`. Ma è possibile mostrare la frazione num:den del numero del gruppo irregolare oppure nascondere del tutto il numero.

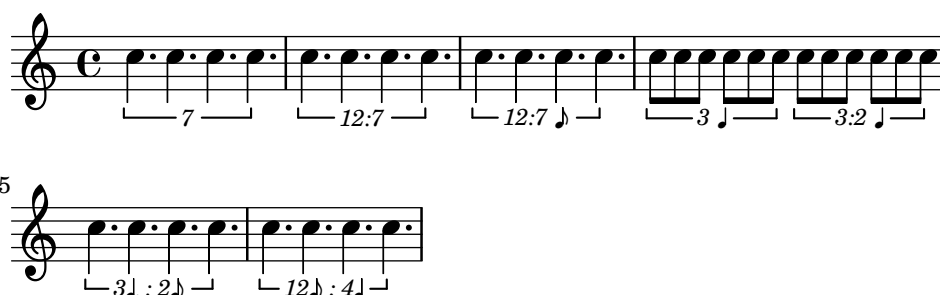
```
\relative c'' {
  \tuplet 3/2 { c8 c c }
  \tuplet 3/2 { c8 c c }
  \override TupletNumber.text = #tuplet-number::calc-fraction-text
  \tuplet 3/2 { c8 c c }
  \omit TupletNumber
  \tuplet 3/2 { c8 c c }
}
```



*Numeri non predefiniti per i gruppi irregolari*

LilyPond fornisce anche funzioni di formattazione che permettono di creare numeri di gruppi irregolari diversi dalla frazione vera e propria, così come di aggiungere un valore di nota al numero o alla frazione di un gruppo irregolare.

```
\relative c'' {
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-tuplet-denominator-text 7)
  \tuplet 3/2 { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-tuplet-fraction-text 12 7)
  \tuplet 3/2 { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      (tuplet-number::non-default-tuplet-fraction-text 12 7) "8")
  \tuplet 3/2 { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      tuplet-number::calc-denominator-text "4")
  \tuplet 3/2 { c8 c8 c8 c8 c8 c8 }
  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      tuplet-number::calc-fraction-text "4")
  \tuplet 3/2 { c8 c8 c8 c8 c8 c8 }
  \once \override TupletNumber.text =
    #(tuplet-number::fraction-with-notes "4." "8")
  \tuplet 3/2 { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-fraction-with-notes 12 "8" 4 "4")
  \tuplet 3/2 { c4. c4. c4. c4. }
}
```

*Controllare la visibilità della parentesi del gruppo irregolare*

Il comportamento predefinito relativo alla visibilità della parentesi quadra del gruppo irregolare è di mostrare una parentesi a meno che non ci sia una travatura della stessa lunghezza del gruppo. Per controllare la visibilità di tale parentesi, si imposta la proprietà 'bracket-visibility su #t (mostra sempre la parentesi), #f (non mostrare mai la parentesi) o #'if-no-beam (mostra la parentesi solo se non c'è una travatura).

```
music = \relative c'' {
  \tuplet 3/2 { c16[ d e ] f8]
  \tuplet 3/2 { c8 d e }
  \tuplet 3/2 { c4 d e }
}
```

```

\new Voice {
  \relative c' {
    << \music s4^"default" >>
    \override TupletBracket.bracket-visibility = #'if-no-beam
    << \music s4^"'if-no-beam" >>
    \override TupletBracket.bracket-visibility = ##t
    << \music s4^"#t" >>
    \override TupletBracket.bracket-visibility = ##f
    << \music s4^"#f" >>
    %% v2.18 :
    \omit TupletBracket
    << \music s4^"omit" >>
  }
}

```



*Consentire l'interruzione del rigo all'interno di gruppi irregolari con travature*

Questo esempio artificioso mostra come permettere interruzioni del rigo sia manuali che automatiche all'interno di un gruppo irregolare con travature. Si noti che le travature di questi gruppi irregolari fuori dal ritmo devono essere disposte manualmente.

```

\layout {
  \context {
    \Voice
    % Permit line breaks within tuplets
    \remove "Forbid_line_break_engraver"
    % Allow beams to be broken at line breaks
    \override Beam.breakable = ##t
  }
}

```

```

}
\relative c'' {
  a8
  \repeat unfold 5 { \tuplet 3/2 { c[ b a] } }
  % Insert a manual line break within a tuplet
  \tuplet 3/2 { c[ b \bar "" \break a] }
  \repeat unfold 5 { \tuplet 3/2 { c[ b a] } }
  c8
}

```



## Vedi anche

Glossario musicale: Sezione “terzina” in *Glossario Musicale*, Sezione “gruppo irregolare” in *Glossario Musicale*, Sezione “polimetrico” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Metodi di modifica” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.4.2 [Direction and placement], pagina 611, [\[Time administration\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [Scaling durations], pagina [\[undefined\]](#), [\[undefined\]](#) [The tweak command], pagina [\[undefined\]](#), [\[undefined\]](#) [Polymetric notation], pagina [\[undefined\]](#).

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TupletBracket” in *Guida al Funzionamento Interno*, Sezione “TupletNumber” in *Guida al Funzionamento Interno*, Sezione “TimeScaledMusic” in *Guida al Funzionamento Interno*.

## Scalare le durate

La durata di singole note, pause o accordi può essere moltiplicata per una frazione  $N/M$  aggiungendo  $*N/M$  (o  $*N$  se  $M$  è 1). Questo non cambierà l’aspetto delle note o delle pause, ma la durata così alterata verrà utilizzata per calcolare la posizione all’interno della misura e per impostare la durata nel file MIDI. Si possono combinare molteplici fattori, come  $*L*M/N$ . I fattori fanno parte della durata: quindi se non si specifica una durata per le note successive, la durata ripresa dalla nota precedente includerà il fattore di scalatura.

Nell’esempio seguente le prime tre note occupano esattamente due tempi, ma non sono indicate come gruppo irregolare.

```

\relative {
  \time 2/4
  % Trasforma le durate in terzine
  a'4*2/3 gis a
  % Durate normali
  a4 a
  % Raddoppia la durata dell'accordo
  <a d>4*2

```

```
% Durata di un quarto, ma appare come un sedicesimo
b16*4 c4
}
```



Anche la durata delle pause spaziatrici può essere modificata con un moltiplicatore. Può essere utile per saltare molte misure; per esempio `s1*23`.

Frammenti musicali più lunghi possono essere compressi secondo la stessa proporzione, come moltiplicando ogni nota, accordo o pausa per una medesima frazione. In questo modo, l'aspetto della musica non cambia ma la durata interna delle note viene moltiplicata per la frazione *num/den*. Ecco un esempio che mostra come la musica possa essere compressa e espansa:

```
\relative {
  \time 2/4
  % Durate normali
  <c' a>4 c8 a
  % Scala la musica di *2/3
  \scaleDurations 2/3 {
    <c a f>4. c8 a f
  }
  % Scala la musica di *2
  \scaleDurations 2/1 {
    <c' a>4 c8 b
  }
}
```



Questo comando torna utile nella notazione polimetrica, si veda [\[Polymetric notation\]](#), pagina [\[Polymetric notation\]](#).

## Vedi anche

Guida alla notazione: [\[Tuplets\]](#), pagina [\[Tuplets\]](#), [\[Invisible rests\]](#), pagina [\[Invisible rests\]](#), [\[Polymetric notation\]](#), pagina [\[Polymetric notation\]](#).

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

## Problemi noti e avvertimenti

Il calcolo della posizione in una misura deve considerare tutti i fattori di dimensionamento applicati alle note di quella misura e gli esigui residui delle misure precedenti. Questo calcolo viene fatto con numeri razionali. Se un numeratore o un denominatore intermedi in quel calcolo eccedono di  $2^{30}$ , l'esecuzione e la composizione tipografica si arresteranno in quel punto senza indicare un errore.

## Legature di valore

Una legatura di valore connette le teste di due note della stessa altezza successive. Dunque, la legatura di valore prolunga la durata di una nota.

**Nota:** Le legature di valore non devono essere confuse con le *legature di portamento*, che articolano un passaggio, o con le *legature di frase*, che delimitano una frase musicale. Una legatura di valore serve semplicemente a prolungare la durata di una nota, in modo analogo al punto di valore.

La legatura di valore si inserisce aggiungendo il simbolo tilde (~) alla prima di ogni coppia di note legate. Esso indica che la nota deve essere legata alla nota successiva, che deve essere della stessa altezza.

```
{ a'2~ 4~ 16 r r8 }
```



Le legature di valore possono avvantaggiarsi dell'interpretazione dell' 'ultima altezza esplicita' per le durate isolate:

```
{ a'2~ 4~ 16 r r8 }
```



Le legature di valore si usano per unire due note a cavallo di una stanghetta di battuta, oppure quando non si possono usare i punti per esprimere una particolare durata. Le legature si dovrebbero usare anche per unire note dalle durate superiori all'unità di suddivisione della misura:

```
\relative {
  r8 c'~ 2 r4 |
  r8^"non" c2~ 8 r4
}
```



Per legare una successione di note la cui durata si prolunga per più misure intere, è più semplice ricorrere alla suddivisione automatica delle note, come è spiegato in [Automatic note splitting](#), pagina [Automatic note splitting](#). Questo metodo divide automaticamente le note lunghe e le connette da misura a misura.

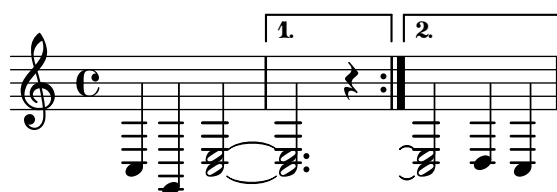
Quando si applica una legatura di valore a degli accordi, vengono legate tutte le teste delle note della stessa altezza. In assenza di altezze corrispondenti, non verrà creata alcuna legatura. Singoli suoni degli accordi possono essere legati inserendo la legatura all'interno dell'accordo stesso.

```
\relative c' {
  <c e g>2~ 2 |
  <c e g>4~ <c e g c>
  <c~ e g~ b> <c e g b> |
}
```



Quando la battuta della "seconda volta" di un ritornello inizia con una nota legata a quella precedente, occorre indicare la legatura nel modo seguente:

```
\relative {
  \repeat volta 2 { c g <c e>2~ }
  \alternative {
    % Prima volta: la nota seguente viene legata in modo normale
    { <c e>2. r4 }
    % Seconda volta: la nota seguente ha una legatura ripetuta
    { <c e>2\repeatTie d4 c }
  }
}
```



Le legature *L.v.* (*laissez vibrer*) indicano che le note non devono essere terminate nettamente. Si usa nella notazione per pianoforte, arpa e altri strumenti a corda e a percussione. Si inseriscono così:

```
<c' f' g'>1\laissezVibrer
```



Le legature di valore possono essere impostate manualmente per avere la curva in su o in giù, come è spiegato in Sezione 5.4.2 [Direction and placement], pagina 611.

Le legature di valore possono essere tratteggiate, punteggiate, oppure tracciate secondo una successione di tratti continui e tratti interrotti.

```
\relative c' {
  \tieDotted
  c2~ 2
  \tieDashed
  c2~ 2
  \tieHalfDashed
  c2~ 2
  \tieHalfSolid
  c2~ 2
  \tieSolid
  c2~ 2
}
```



Si possono specificare modelli di tratteggiatura personalizzati:

```
\relative c' {
```



```

\tieDashPattern #0.3 #0.75
c2~ 2
\tieDashPattern #0.7 #1.5
c2~ 2
\tieSolid
c2~ 2
}

```



Le definizioni dei modelli di tratteggiatura delle legature di valore hanno la stessa struttura di quelle per le legature di portamento. I dettagli relativi ai modelli complessi di tratteggiatura sono trattati in [\[Slurs\]](#), pagina [\[Slurs\]](#).

Sovrascrivere le proprietà di formattazione *whiteout* e *layer* degli oggetti che devono creare uno spazio vuoto tra le legature di valore.

```

\relative {
  \override Tie.layer = #-2
  \override Staff.TimeSignature.layer = #-1
  \override Staff.KeySignature.layer = #-1
  \override Staff.TimeSignature.whiteout = ##t
  \override Staff.KeySignature.whiteout = ##t
  b'2 b~
  \time 3/4
  \key a \major
  b r4
}

```



## Comandi predefiniti

`\tieUp`, `\tieDown`, `\tieNeutral`, `\tieDotted`, `\tieDashed`, `\tieDashPattern`, `\tieHalfDashed`, `\tieHalfSolid`, `\tieSolid`.

## Frammenti di codice selezionati

*Usare le legature di valore con un arpeggio*

Le legature di valore vengono usate talvolta per scrivere un arpeggio. In questo caso, le due note da legare devono non essere consecutive. Per ottenere tale risultato occorre impostare la proprietà `tieWaitForNote` su `##t`. Questa funzionalità serve anche a legare un tremolo a un accordo e in generale qualsiasi coppia di note consecutive.

```

\relative c' {
  \set tieWaitForNote = ##t
  \grace { c16[ ~ e ~ g] ~ } <c, e g>2
  \repeat tremolo 8 { c32 ~ c' ~ } <c c,>1
  e8 ~ c ~ a ~ f ~ <e' c a f>2
  \tieUp
  c8 ~ a
}

```

```

\tieDown
\tieDotted
g8 ~ c g2
}

```



*Disegnare manualmente le legature di valore*

Le legature di valore possono essere disegnate a mano cambiando la proprietà `tie-configuration` dell'oggetto `TieColumn`. Il primo numero indica la distanza dal centro del rigo nell'unità di metà spazio rigo, mentre il secondo numero indica la direzione (1 = su, -1 = giù).

```

\relative c' {
  <c e g>2~ <c e g>
  \override TieColumn.tie-configuration =
    #'((0.0 . 1) (-2.0 . 1) (-4.0 . 1))
  <c e g>2~ <c e g>
}

```



## Vedi anche

Glossario musicale: Sezione “legatura di valore” in *Glossario Musicale*, Sezione “laissez vibrer” in *Glossario Musicale*.

Guida alla notazione: `<undefined>` [Slurs], pagina `<undefined>`, `<undefined>` [Automatic note splitting], pagina `<undefined>`.

Frammenti di codice: Sezione “Expressive marks” in *Frammenti di codice*, Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “LaissezVibrerTie” in *Guida al Funzionamento Interno*, Sezione “LaissezVibrerTieColumn” in *Guida al Funzionamento Interno*, Sezione “TieColumn” in *Guida al Funzionamento Interno*, Sezione “Tie” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Cambiare rigo mentre una legatura di valore è attiva non produce una legatura obliqua.

Il cambio di chiave o di ottava durante una legatura di valore non è una situazione ben definita. In questi casi è preferibile usare una legatura di portamento.

### 1.2.2 Inserimento delle pause

Le pause si inseriscono insieme alla musica contenuta nelle espressioni musicali.

#### Pause

Le pause si inseriscono allo stesso modo delle note, ma con il carattere `r`. Le durate più lunghe di un intero usano i seguenti comandi predefiniti:

```

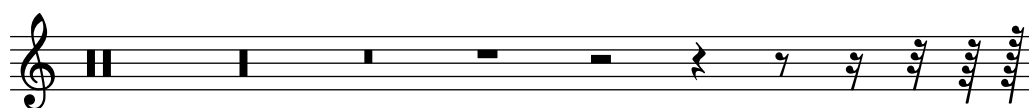
\new Staff {

```

```

% Queste due linee servono solo ad abbellire questo esempio
\time 16/1
\omit Staff.TimeSignature
% Mostra una pausa di maxima, equivalente a quattro brevi
r\maxima
% Mostra una pausa di longa, equivalente a due brevi
r\longa
% Mostra una pausa di breve
r\breve
r1 r2 r4 r8 r16 r32 r64 r128
}

```



Le pause d'intero, poste al centro della misura, devono essere inserite come pause multiple. Si possono usare sia per una sola misura sia su più misure, come è spiegato in [\[Full measure rests\]](#), pagina [\[Full measure rests\]](#).

Per indicare esplicitamente la posizione verticale di una pausa, si scrive la nota corrispondente seguita da `\rest`. Una pausa della durata della nota verrà collocata nella posizione della nota sul rigo. Questo permette una precisa formattazione manuale della musica polifonica, dato che il formattatore automatico che gestisce le collisioni tra pause non interviene su questo tipo di pause.

```
\relative { a'4\rest d4\rest }
```



## Frammenti di codice selezionati

### *Stili di pausa*

Esistono vari stili di pausa.

```

\new Staff \relative c {
  \omit Score.TimeSignature
  \cadenzaOn

  \override Staff.Rest.style = #'mensural
  r\maxima^{\markup \typewriter { mensural }}
  r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
  \bar ""
  \break

  \override Staff.Rest.style = #'neomensural
  r\maxima^{\markup \typewriter { neomensural }}
  r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
  \bar ""
  \break

  \override Staff.Rest.style = #'classical

```

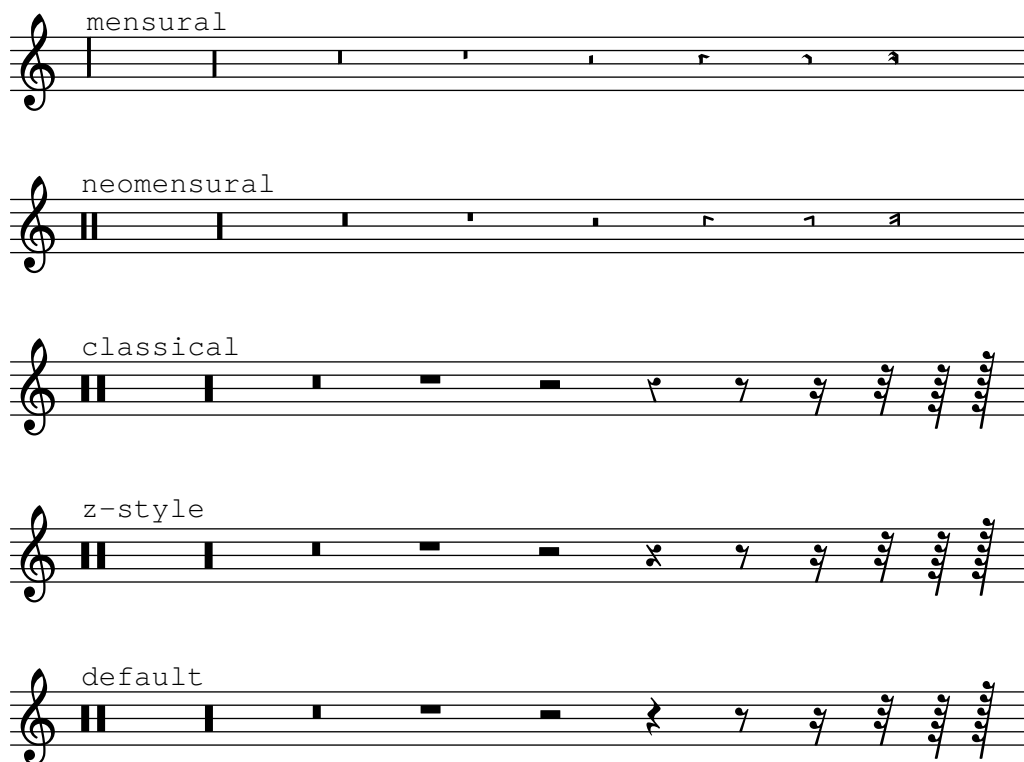
```

r\maxima^\markup \typewriter { classical }
r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 s128
\bar ""
\break

\override Staff.Rest.style = #'z
r\maxima^\markup \typewriter { z-style }
r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 s128
\bar ""
\break

\override Staff.Rest.style = #'default
r\maxima^\markup \typewriter { default }
r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 s128
}

```



## Vedi anche

Glossario musicale: Sezione “breve” in *Glossario Musicale*, Sezione “longa” in *Glossario Musicale*, Sezione “maxima” in *Glossario Musicale*.

Guida alla notazione: [\[Full measure rests\]](#), pagina [\[Full measure rests\]](#).

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Rest” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Non c'è un limite massimo o minimo alla durata di una pausa, ma è il numero dei glifi ad essere limitato: si possono indicare pause da un centoventottesimo fino alla maxima (otto volte una semibreve).

## Pause invisibili

Una pausa invisibile (chiamata anche ‘pausa spaziatrice’) si inserisce come una nota col nome `s`:

```
\relative c'' {
  c4 c s c |
  s2 c |
}
```



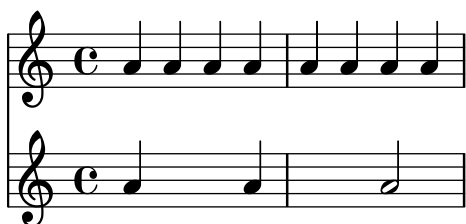
Le pause spaziatrici possono essere usate soltanto nella modalità note e nella modalità accordi. In altre situazioni, ad esempio quando si inserisce il testo vocale, si usa il comando `\skip` per saltare un valore musicale. `\skip` richiede una durata esplicita, ma questo requisito viene ignorato se il testo desume le proprie durate dalle note presenti in una melodia ad esso associata attraverso `\addlyrics` o `\lyricsto`.

```
<<
{
  a'2 \skip2 a'2 a'2
}
\new Lyrics {
  \lyricmode {
    foo2 \skip 1 bla2
  }
}
>>
```



Dato che `\skip` è un comando, non modifica la durata predefinita delle note che seguono, diversamente da `s`.

```
<<
{
  \repeat unfold 8 { a'4 }
}
{
  a'4 \skip 2 a' |
  s2 a'
}
>>
```



Una pausa spaziatrice crea implicitamente i contesti **Staff** e **Voice** se non esistono già, proprio come accade per le note e le pause:

```
{ s1 s s }
```



`\skip` si limita a saltare un valore musicale, non crea nessun tipo di output.

```
% Questo input è corretto, ma non produce niente
\skip 1 \skip1 \skip 1
```

## Vedi anche

Manuale di apprendimento: Sezione “Visibilità e colore degli oggetti” in *Manuale di Apprendimento*.

Guida alla notazione: `\hidden` [Hidden notes], pagina `\hidden`, Sezione 5.4.7 [Visibility of objects], pagina 618.

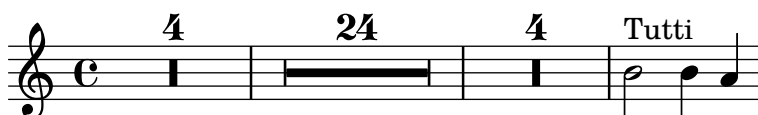
Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “SkipMusic” in *Guida al Funzionamento Interno*.

## Pause d'intero

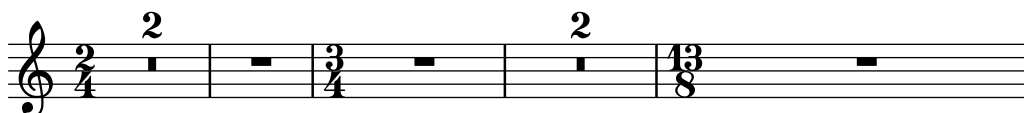
Le pause per una o più misure d'intero si inseriscono, come le note, col carattere maiuscolo **R**:

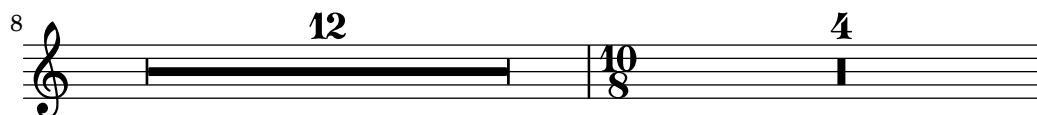
```
% L'insieme delle misure di pausa vengono riportate in una sola misura
\compressMMRests {
  R1*4
  R1*24
  R1*4
  b'2~"Tutti" b'4 a'4
}
```



La durata delle pause multiple è identica alla notazione di durata usata per le note e deve essere sempre un numero intero di misure/lunghezze, quindi occorre spesso usare dei punti di aumentazione o delle frazioni:

```
\compressMMRests {
  \time 2/4
  R1 | R2 |
  \time 3/4
  R2. | R2.*2 |
  \time 13/8
  R1*13/8 | R1*13/8*12 |
  \time 10/8
  R4*5*4 |
}
```





Una pausa d'intero appare al centro della misura con la durata di una semibreve o di una breve, in base all'indicazione di tempo.

```
\time 4/4
R1 |
\time 6/4
R1*3/2 |
\time 8/4
R1*2 |
```



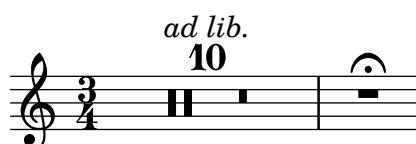
Di norma una pausa multipla viene scorporata sul pentagramma in modo da mostrare esplicitamente tutte le misure per cui si prolunga. Altrimenti, è possibile indicarla collocando in una sola misura un simbolo di pausa multipla, col numero di misure per cui la pausa si prolunga posto al di sopra della misura stessa:

```
% Comportamento predefinito
\time 3/4 r2. | R2.*2 |
\time 2/4 R2 |
\time 4/4
% Tutte le misure di pausa sono riportate in una singola misura
\compressMMRests {
  r1 | R1*17 | R1*4 |
}
% Le misure della pausa multipla sono scorporate di nuovo
\time 3/4
R2.*2 |
```



Si possono aggiungere delle annotazioni alle pause multiple. Il comando predefinito `\fermataMarkup` permette di aggiungere il segno di corona.

```
\compressMMRests {
  \time 3/4
  R2.*10^\markup { \italic "ad lib." }
  R2.\fermataMarkup
}
```



**Nota:** Il testo connesso a una pausa multipla è un oggetto di tipo `MultiMeasureRestText`, non `TextScript`. Le sovrascritture devono specificare l'oggetto corretto o saranno ignorate. Si veda l'esempio seguente:

```
% Questo non funziona, perché è specificato il nome dell'oggetto sbagliato
\override TextScript.padding = #5
R1^"sbagliato"
% Questo è il nome dell'oggetto corretto da specificare
\override MultiMeasureRestText.padding = #5
R1^"corretto"
```



Quando una pausa multipla segue immediatamente un comando `\partial`, potrebbero non apparire i relativi avvertimenti del controllo battuta.

## Comandi predefiniti

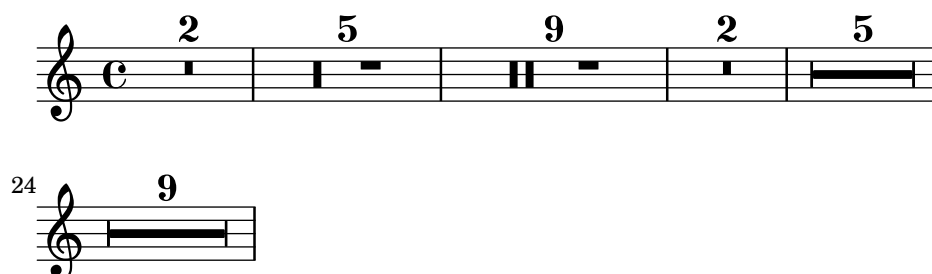
`\textLengthOn`, `\textLengthOff`, `\fermataMarkup`, `\compressMMRests`,

## Frammenti di codice selezionati

*Modificare la forma delle pause multiple*

Se la pausa multipla dura dieci misure o un numero inferiore a dieci, nel rigo apparirà una serie di pause di lunga e di breve (chiamate in tedesco “Kirchenpausen” - pause ecclesiastiche); altrimenti apparirà una semplice linea. Il numero predefinito di dieci può essere cambiato sovrascrivendo la proprietà `expand-limit`.

```
\relative c'' {
  \compressMMRests {
    R1*2 | R1*5 | R1*9
    \override MultiMeasureRest.expand-limit = #3
    R1*2 | R1*5 | R1*9
  }
}
```



*Posizionamento delle pause multiple*

Diversamente dalle pause normali, non esiste un comando predefinito per cambiare la posizione sul rigo di un simbolo di pausa multipla di qualsiasi tipo connettendolo a una nota. Tuttavia, nella musica polifonica le pause multiple nelle voci dispari e pari sono separate verticalmente. Il posizionamento delle pause multiple si controlla nel modo seguente:

```
\relative c'' {
```



```

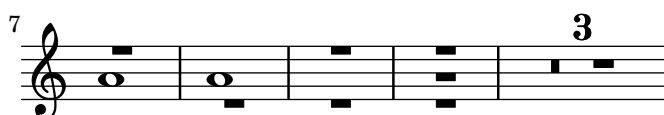
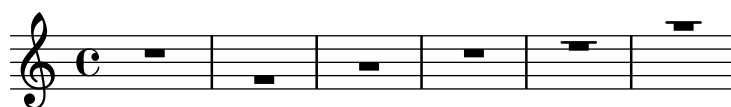
% Multi-measure rests by default are set under the fourth line
R1
% They can be moved using an override
\override MultiMeasureRest.staff-position = #-2
R1
\override MultiMeasureRest.staff-position = #0
R1
\override MultiMeasureRest.staff-position = #2
R1
\override MultiMeasureRest.staff-position = #3
R1
\override MultiMeasureRest.staff-position = #6
R1
\revert MultiMeasureRest.staff-position
\break

% In two Voices, odd-numbered voices are under the top line
<< { R1 } \\\ { a1 } >>
% Even-numbered voices are under the bottom line
<< { a1 } \\\ { R1 } >>
% Multi-measure rests in both voices remain separate
<< { R1 } \\\ { R1 } >>

% Separating multi-measure rests in more than two voices
% requires an override
<< { R1 } \\\ { R1 } \\\
  \once \override MultiMeasureRest.staff-position = #0
  { R1 }
>>

% Using compressed bars in multiple voices requires another override
% in all voices to avoid multiple instances being printed
\compressMMRests
<<
  \revert MultiMeasureRest.direction
  { R1*3 }
  \\\
  \revert MultiMeasureRest.direction
  { R1*3 }
>>
}

```

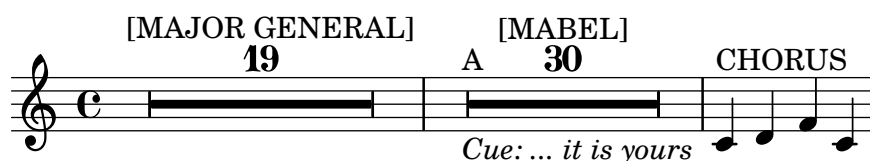


*Testo a margine delle pause multiple*

Il testo a margine di una pausa multipla viene centrato sopra o sotto di essa. Se il testo è lungo, la misura non si espanderà. Per espandere la pausa multipla in modo che si allinei col testo, conviene usare un accordo vuoto con del testo attaccato prima della pausa multipla.

Il testo così attaccato a una nota spaziatrice viene allineato a sinistra della posizione in cui la nota sarebbe posta nella misura, ma se la lunghezza della misura è determinata dalla lunghezza del testo, il testo verrà centrato.

```
\relative c' {
  \compressMMRests {
    \textLengthOn
    <>\markup { [MAJOR GENERAL] }
    R1*19
    <>\markup { \italic { Cue: ... it is yours } }
    <>\markup { A }
    R1*30\markup { [MABEL] }
    \textLengthOff
    c4\markup { CHORUS } d f c
  }
}
```

**Vedi anche**

Glossario musicale: Sezione “pausa multipla” in *Glossario Musicale*.

Guida alla notazione: [\[Durations\]](#), pagina [\[Text\]](#), pagina [\[Formatting text\]](#), pagina [\[Text scripts\]](#), pagina [\[Text scripts\]](#).

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “MultiMeasureRest” in *Guida al Funzionamento Interno*, Sezione “MultiMeasureRestNumber” in *Guida al Funzionamento Interno*, Sezione “MultiMeasureRestText” in *Guida al Funzionamento Interno*.

**Problemi noti e avvertimenti**

Se una ditekgiatura viene posta su una pausa multipla (ad esempio `R1*10-4`), il numero della ditekgiatura può collidere col numero del contatore delle battute.

Non è possibile condensare automaticamente molteplici pause normali in in una singola pausa multipla.

Le pause multiple non considerano le collisioni di pausa.

**1.2.3 Aspetto dei ritmi****Indicazione di tempo**

L’indicazione di tempo si imposta così:

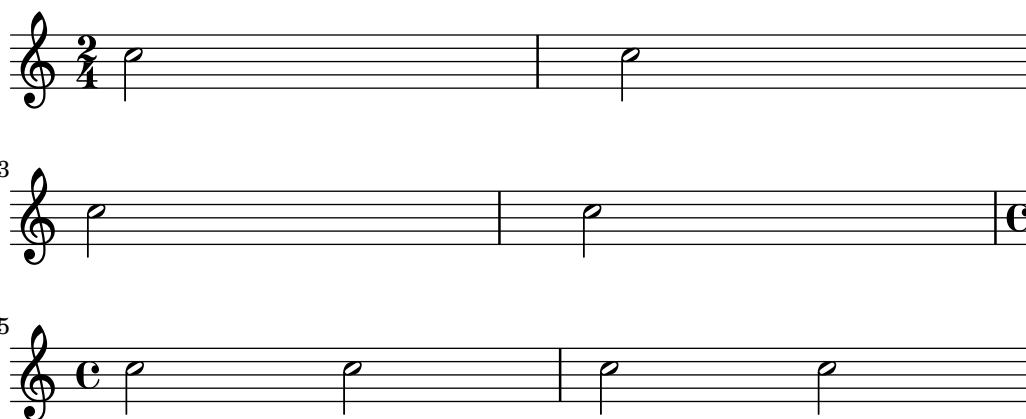
```
\time 2/4 c''2
\time 3/4 c''2.
```



I cambi di indicazione di tempo a metà misura sono trattati in [\[Upbeats\]](#), pagina [\[undefined\]](#).

Le indicazioni di tempo appaiono all'inizio di un brano e ogni volta che l'indicazione cambia. Se il cambio ha luogo alla fine di un rigo, appare un'indicazione di tempo di precauzione. Si può modificare questo comportamento predefinito, come è spiegato in Sezione 5.4.7 [\[Visibility of objects\]](#), pagina 618.

```
\relative c'' {
  \time 2/4
  c2 c
  \break
  c c
  \break
  \time 4/4
  c c c c
}
```



Il simbolo di indicazione di tempo usato nei tempi 2/2 e 4/4 può essere sostituito da un numero:

```
\relative c'' {
  % Stile predefinito
  \time 4/4 c1
  \time 2/2 c1
  % Passaggio allo stile numerico
  \numericTimeSignature
  \time 4/4 c1
  \time 2/2 c1
  % Ritorno allo stile predefinito
  \defaultTimeSignature
  \time 4/4 c1
  \time 2/2 c1
}
```



Le indicazioni di tempo mensurali sono trattate in [\[Mensural time signatures\]](#), pagina 437.

Oltre a impostare l'indicazione di tempo che appare nel pentagramma, il comando `\time` imposta anche i valori delle proprietà basate sull'indicazione di tempo, ovvero `baseMoment`, `beatStructure` e `beamExceptions`. I valori predefiniti di queste proprietà si trovano in `scm/time-signature-settings.scm`.

Si può sovrascrivere il valore predefinito di `beatStructure` nel comando `\time` stesso specificandolo come primo argomento opzionale:

```
\score {
  \new Staff {
    \relative {
      \time #'(2 2 3) 7/8
      \repeat unfold 7 { c'8 } |
      \time #'(3 2 2) 7/8
      \repeat unfold 7 { c8 } |
    }
  }
}
```



Oppure si possono impostare tutti i valori predefiniti di queste variabili relative all'indicazione di tempo, incluse `baseMoment` e `beamExceptions`. I valori possono essere impostati in modo indipendente per diverse indicazioni di tempo. I nuovi valori hanno effetto appena viene eseguito un nuovo comando `\time` che abbia lo stesso valore dell'indicazione di tempo specificata nelle nuove impostazioni:

```
\score {
  \new Staff {
    \relative c' {
      \overrideTimeSignatureSettings
        4/4          % timeSignatureFraction
        1/4          % baseMomentFraction
        #'(3 1)      % beatStructure
        #'()         % beamExceptions
      \time 4/4
      \repeat unfold 8 { c8 } |
    }
  }
}
```



`\overrideTimeSignatureSettings` prende quattro argomenti:

1. *timeSignatureFraction*, una frazione che indica l'indicazione di tempo a cui questi valori si riferiscono.
2. *baseMomentFraction*, una frazione che contiene il numeratore e il denominatore dell'unità di tempo.
3. *beatStructure*, una lista Scheme che indica la struttura dei battiti nella misura, nell'unità di *baseMomentFraction*.

4. *beamExceptions*, una lista di associazione (*alist*) che contiene regole di disposizione delle travature che vanno oltre la fine ad ogni battuto, come descritto in [\[Setting automatic beam behavior\]](#), pagina [\(undefined\)](#).

I valori modificati delle proprietà predefinite dell'indicazione di tempo possono essere ripristinati ai valori originali:

```
\score{
  \relative {
    \repeat unfold 8 { c'8 } |
    \overrideTimeSignatureSettings
      4/4      % timeSignatureFraction
      1/4      % baseMomentFraction
      #'(3 1)  % beatStructure
      #'()     % beamExceptions
    \time 4/4
    \repeat unfold 8 { c8 } |
    \revertTimeSignatureSettings 4/4
    \time 4/4
    \repeat unfold 8 { c8 } |
  }
}
```



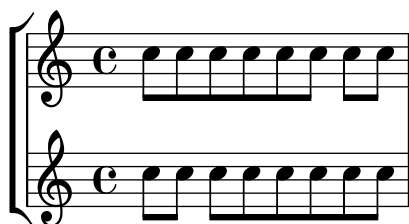
Si possono stabilire valori diversi delle proprietà predefinite dell'indicazione di tempo per righe diversi spostando *Timing\_translator* e *Default\_bar\_line\_engraver* dal contesto *Score* al contesto *Staff*.

```
\score {
  \new StaffGroup <<
    \new Staff {
      \overrideTimeSignatureSettings
        4/4      % timeSignatureFraction
        1/4      % baseMomentFraction
        #'(3 1)  % beatStructure
        #'()     % beamExceptions
      \time 4/4
      \repeat unfold 8 {c''8}
    }
    \new Staff {
      \overrideTimeSignatureSettings
        4/4      % timeSignatureFraction
        1/4      % baseMomentFraction
        #'(1 3)  % beatStructure
        #'()     % beamExceptions
      \time 4/4
      \repeat unfold 8 {c''8}
    }
  }
  >>
  \layout {
    \context {
```

```

\Score
\remove "Timing_translator"
\remove "Default_bar_line_engraver"
}
\context {
  \Staff
  \consists "Timing_translator"
  \consists "Default_bar_line_engraver"
}
}
}

```



Un ulteriore metodo per modificare queste variabili relative all'indicazione di tempo, che evita di mostrare di nuovo l'indicazione di tempo al momento del cambio, è descritto in [\[Setting automatic beam behavior\]](#), pagina [\[undefined\]](#).

## Comandi predefiniti

`\numericTimeSignature`, `\defaultTimeSignature`.

## Frammenti di codice selezionati

*Indicazione di tempo che mostra solo il numeratore (invece della frazione)*

Talvolta un'indicazione di tempo non deve mostrare la frazione intera (ad esempio 7/4), ma solo il numeratore (7 in questo caso). Si può ottenere facilmente con `\override Staff.TimeSignature.style = #'single-digit`, che cambia lo stile in modo permanente. Con `\revert Staff.TimeSignature.style`, questa impostazione può essere annullata. Per applicare lo stile a cifra singola (`single-digit`) a una sola indicazione di tempo, si usa il comando `\override` preceduto da `\once`.

```

\relative c'' {
  \time 3/4
  c4 c c
  % Change the style permanently
  \override Staff.TimeSignature.style = #'single-digit
  \time 2/4
  c4 c
  \time 3/4
  c4 c c
  % Revert to default style:
  \revert Staff.TimeSignature.style
  \time 2/4
  c4 c
  % single-digit style only for the next time signature
  \once \override Staff.TimeSignature.style = #'single-digit
  \time 5/4
  c4 c c c c
}

```

```
\time 2/4
c4 c
}
```



## Vedi anche

Glossario musicale: Sezione “indicazione di tempo” in *Glossario Musicale*

Guida alla notazione: [Mensural time signatures], pagina 437, (undefined) [Setting automatic beam behavior], pagina (undefined), (undefined) [Time administration], pagina (undefined).

File installati: `scm/time-signature-settings.scm`.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TimeSignature” in *Guida al Funzionamento Interno*, Sezione “Timing-translator” in *Guida al Funzionamento Interno*.

## Indicazioni metronomiche

Un’indicazione metronomica è semplice da scrivere:

```
\relative {
  \tempo 4 = 120
  c'2 d
  e4. d8 c2
}
```



Le indicazioni metronomiche si possono rappresentare anche come una gamma di due numeri:

```
\relative {
  \tempo 4 = 40 - 46
  c'4. e8 a4 g
  b,2 d4 r
}
```



Al loro posto si possono usare delle indicazioni di tempo testuali:

```
\relative {
  \tempo "Allegretto"
  c'4 e d c
  b4. a16 b c4 r4
}
```



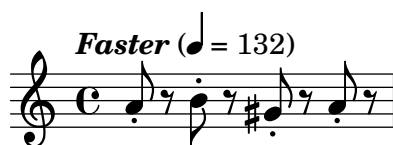
Un'indicazione metronomica, se combinata con del testo, viene posta automaticamente tra parentesi:

```
\relative {
  \tempo "Allegro" 4 = 160
  g'4 c d e
  d4 b g2
}
```



In generale, il testo può essere qualsiasi oggetto di tipo testuale:

```
\relative {
  \tempo \markup { \italic Faster } 4 = 132
  a'8-. r8 b-. r gis-. r a-. r
}
```



È possibile scrivere un'indicazione metronomica tra parentesi e senza testo includendo una stringa vuota nell'input:

```
\relative {
  \tempo "" 8 = 96
  d'4 g e c
}
```

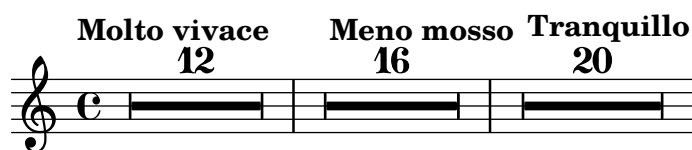


In una parte per uno strumento che ha lunghi periodi pieni di pause, le indicazioni di tempo sono talvolta molto ravvicinate. Il comando `\markLengthOn` aggiunge dello spazio orizzontale per impedire che le indicazioni di tempo si sovrappongano; `\markLengthOff` ripristina il comportamento predefinito, per cui le indicazioni di tempo non sono tenute in considerazione ai fini della spaziatura orizzontale.

```
\compressMMRests {
  \markLengthOn
  \tempo "Molto vivace"
  R1*12
  \tempo "Meno mosso"
  R1*16
  \markLengthOff
  \tempo "Tranquillo"
  R1*20
}
```



}



## Frammenti di codice selezionati

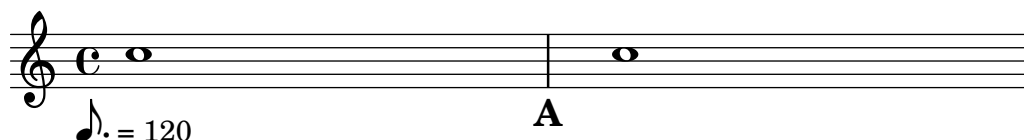
*Posizionare il metronomo e i numeri di chiamata sotto il rigo*

Di norma, il metronomo e i numeri di chiamata vengono posizionati sopra il rigo. Per metterli sotto il rigo basta impostare correttamente la proprietà `direction` di `MetronomeMark` o `RehearsalMark`.

```
\layout {
  indent = 0
  ragged-right = ##f
}

{
  % Metronome marks below the staff
  \override Score.MetronomeMark.direction = #DOWN
  \tempo 8. = 120
  c''1

  % Rehearsal marks below the staff
  \override Score.RehearsalMark.direction = #DOWN
  \mark \default
  c''1
}
```



*Modificare il tempo senza mostrare l'indicazione metronomica*

Per cambiare il tempo del file MIDI senza che appaia l'indicazione metronomica, basta renderla invisibile.

```
\score {
  \new Staff \relative c' {
    \tempo 4 = 160
    c4 e g b
    c4 b d c
    \set Score.tempohideNote = ##t
    \tempo 4 = 96
    d,4 fis a cis
    d4 cis e d
  }
  \layout { }
  \midi { }
}
```



### Creare indicazioni metronomiche in modalità testuale

Si possono creare nuove indicazioni metronomiche in modalità testuale, ma non modificheranno il tempo del file MIDI.

```
\relative c' {
  \tempo \markup {
    \concat {
      (
        \smaller \general-align #Y #DOWN \note #"16." #1
        " = "
        \smaller \general-align #Y #DOWN \note #"8" #1
      )
    }
  }
  c1
  c4 c' c,2
}
```



I dettagli si trovano in [\[Formatting text\]](#), pagina [\[undefined\]](#).

## Vedi anche

Glossario musicale: Sezione “metronomo” in *Glossario Musicale*, Sezione “indicazione di tempo” in *Glossario Musicale*, Sezione “indicazione metronomica” in *Glossario Musicale*.

Guida alla notazione: [\[undefined\]](#) [\[Formatting text\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Creating MIDI output\]](#), pagina [\[undefined\]](#).

Frammenti di codice: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “MetronomeMark” in *Guida al Funzionamento Interno*.

## Anacrusi

Le misure parziali, come l'*anacrusi* o la battuta in levare, si inseriscono col comando `\partial`:

```
\partial durata
```

Quando si usa `\partial` all'inizio di una partitura, la *durata* è la lunghezza della musica che precede la prima battuta.

```
\relative {
  \time 3/4
  \partial 4.
  r4 e'8 | a4 c8 b c4 |
}
```



Quando si usa `\partial` dopo l'inizio di una partitura, la *durata* è la lunghezza *rimanente* della misura corrente. Non crea una battuta con un nuovo numero.

```
\relative {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \override Score.BarNumber.break-visibility =
    #end-of-line-invisible
  \time 9/8
  d''4.~ 4 d8 d( c) b | c4.~ 4. \bar "||"
  \time 12/8
  \partial 4.
  c8( d) e | f2.~ 4 f8 a,( c) f |
}
```



Il comando `\partial` è *obbligatorio* quando l'indicazione di tempo cambia in mezzo a una misura, ma si può usare anche da solo.

```
\relative {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \override Score.BarNumber.break-visibility =
    #end-of-line-invisible
  \time 6/8
  \partial 8
  e'8 | a4 c8 b[ c b] |
  \partial 4
  r8 e,8 | a4 \bar "||"
  \partial 4
  r8 e8 | a4
  c8 b[ c b] |
}
```



Il comando `\partial` imposta la proprietà `Timing.measurePosition`, che è un numero razionale che indica quanto tempo della misura è trascorsa.

## Vedi anche

Glossario musicale: Sezione “anacrusi” in *Glossario Musicale*.

Guida alla notazione: `<undefined>` [Grace notes], pagina `<undefined>`.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Timing-translator” in *Guida al Funzionamento Interno*.

## Musica in tempo libero

Nella musica in un tempo determinato l'inserimento delle stanghette e dei numeri di battuta è calcolato automaticamente. Nella musica in tempo libero (per esempio, la cadenza), un simile comportamento non è desiderabile, e può essere 'disabilitato' col comando `\cadenzaOn` e poi 'riabilitato' quando necessario con `\cadenzaOff`.

```
\relative c'' {
  c4 d e d
  \cadenzaOn
  c4 c d8[ d d] f4 g4.
  \cadenzaOff
  \bar "|"
  d4 e d c
}
```



La numerazione delle battute riprende dopo la cadenza.

```
\relative c'' {
  % Mostra tutti i numeri di battuta
  \override Score.BarNumber.break-visibility = #all-visible
  c4 d e d
  \cadenzaOn
  c4 c d8[ d d] f4 g4.
  \cadenzaOff
  \bar "|"
  d4 e d c
}
```



Se si inserisce un comando `\bar` dentro una cadenza non viene iniziata una nuova misura, anche se appare una stanghetta nell'output. Quindi qualsiasi alterazione, che di solito si considera sempre attiva fino alla fine della misura, sarà ancora valida dopo la stanghetta stampata da `\bar`. Se si desidera che le alterazioni successive appaiano, si dovranno inserire manualmente delle alterazioni forzate o di precauzione, come è spiegato in [\[Accidentals\]](#), pagina [\[undefined\]](#).

```
\relative c'' {
  c4 d e d
  \cadenzaOn
  cis4 d cis d
  \bar "|"
  % Il primo cis viene stampato senza alterazione anche se si trova dopo \bar
  cis4 d cis! d
  \cadenzaOff
  \bar "|"
}
```



La disposizione automatica delle travature viene disabilitata da `\cadenzaOn`. Quindi tutte le travature nelle cadenze devono essere inserite manualmente. Si veda [\[Manual beams\]](#), pagina [\[Manual beams\]](#).

```
\relative {
  \repeat unfold 8 { c''8 }
  \cadenzaOn
  cis8 c c c c
  \bar""|
  c8 c c
  \cadenzaOff
  \repeat unfold 8 { c8 }
}
```



Questi comandi predefiniti hanno effetto su tutti i righi di una partitura, anche quando inseriti in un solo contesto **Voice**. Per modificare questo comportamento, si sposta `Timing_translator` dal contesto **Score** al contesto **Staff**. Si veda [\[Polymetric notation\]](#), pagina [\[Polymetric notation\]](#).

## Comandi predefiniti

`\cadenzaOn`, `\cadenzaOff`.

## Vedi anche

Glossario musicale: Sezione “cadenza” in *Glossario Musicale*.

Guida alla notazione: Sezione 5.4.7 [\[Visibility of objects\]](#), pagina 618, [\[Polymetric notation\]](#), pagina [\[Polymetric notation\]](#), [\[Manual beams\]](#), pagina [\[Manual beams\]](#), [\[Accidentals\]](#), pagina [\[Accidentals\]](#).

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

## Problemi noti e avvertimenti

Le interruzioni automatiche di linea e di pagina possono aver luogo solo dopo una stanghetta di battuta; quindi, per consentire delle interruzioni nei lunghi passaggi di musica in tempo libero è necessario inserire manualmente delle stanghette ‘invisibili’:

```
\bar ""
```

## Notazione polimetrica

La notazione polimetrica è supportata esplicitamente o tramite la modifica manuale del simbolo d’indicazione di tempo (e la trasformazione della durata delle note).

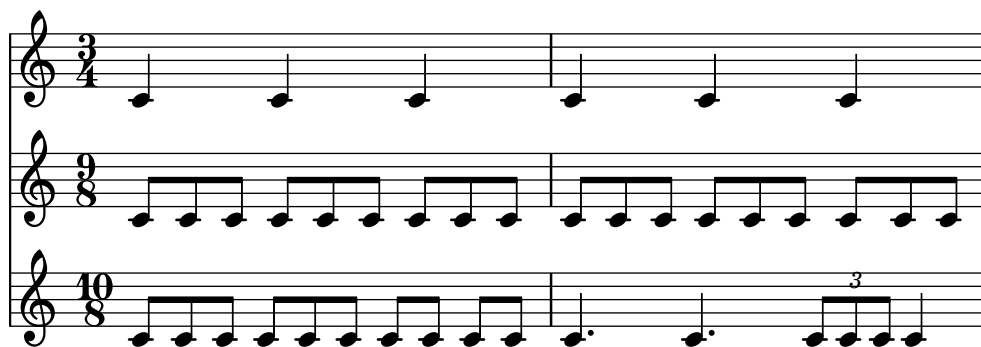
### *Diverse indicazioni di tempo con misure di uguale lunghezza*

Si sceglie una normale indicazione di tempo per ogni rigo e si imposta `timeSignatureFraction` sulla frazione desiderata. Quindi si usa la funzione `\scaleDurations` per scalare la durata delle note di ogni rigo in modo che rientrino nella comune indicazione di tempo.

L’esempio seguente presenta simultaneamente musica con indicazioni di tempo di  $3/4$ ,  $9/8$  e  $10/8$ . Nel secondo rigo le durate appaiono come moltiplicate per  $2/3$  (perché  $2/3 * 9/8 = 3/4$ ), mentre nel terzo rigo le durate appaiono come moltiplicate per  $3/5$  (perché  $3/5 * 10/8 = 3/4$ ). È

possibile che si debbano inserire a mano le travature, perché la scalatura delle durate influenzerà le regole della disposizione automatica delle travature.

```
\relative <<
  \new Staff {
    \time 3/4
    c'4 c c |
    c4 c c |
  }
  \new Staff {
    \time 3/4
    \set Staff.timeSignatureFraction = 9/8
    \scaleDurations 2/3
    \repeat unfold 6 { c8[ c c] }
  }
  \new Staff {
    \time 3/4
    \set Staff.timeSignatureFraction = 10/8
    \scaleDurations 3/5 {
      \repeat unfold 2 { c8[ c c] }
      \repeat unfold 2 { c8[ c] } |
      c4. c \tuplet 3/2 { c8[ c c] } c4
    }
  }
>>
```



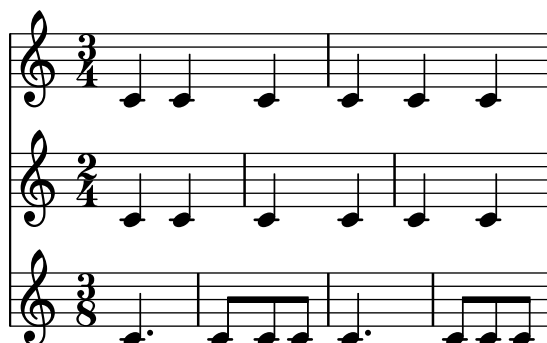
### *Diverse indicazioni di tempo con misure di lunghezza differenti*

Si può dare a ogni rigo la sua indicazione di tempo indipendente spostando `Timing_translator` e `Default_bar_line_engraver` nel contesto `Staff`.

```
\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Default_bar_line_engraver"
  }
  \context {
    \Staff
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  }
}
```

% Ora ogni rigo ha la sua indicazione di tempo.

```
\relative <<
  \new Staff {
    \time 3/4
    c'4 c c |
    c4 c c |
  }
  \new Staff {
    \time 2/4
    c4 c |
    c4 c |
    c4 c |
  }
  \new Staff {
    \time 3/8
    c4. |
    c8 c c |
    c4. |
    c8 c c |
  }
>>
```



### Indicazioni di tempo composto

Si creano con la funzione `\compoundMeter`. La sintassi è:

```
\compoundMeter #'(lista di liste)
```

La struttura più semplice è una singola lista, dove l'*ultimo* numero indica il numero inferiore dell'indicazione di tempo e i numeri precedenti indicano i numeri superiori del segno di tempo.

```
\relative {
  \compoundMeter #'((2 2 2 8))
  \repeat unfold 6 c'8 \repeat unfold 12 c16
}
```



Si possono costruire tempi più complessi tramite ulteriori liste. Le modalità di disposizione automatica delle travature varieranno a seconda di questi valori.

```
\relative {
```

```

\compoundMeter #'((1 4) (3 8))
\repeat unfold 5 c'8 \repeat unfold 10 c16
}

\relative {
  \compoundMeter #'((1 2 3 8) (3 4))
  \repeat unfold 12 c'8
}

```



## Vedi anche

Glossario musicale: Sezione “polimetrico” in *Glossario Musicale*, Sezione “indicazione di tempo polimetrico” in *Glossario Musicale*, Sezione “tempo” in *Glossario Musicale*.

Guida alla notazione: [Automatic beams](#), pagina [Manual beams](#), pagina [Time signature](#), pagina [Scaling durations](#), pagina [Frammenti di codice](#).

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TimeSignature” in *Guida al Funzionamento Interno*, Sezione “Timing\_translator” in *Guida al Funzionamento Interno*, Sezione “Default\_bar\_line\_engraver” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Quando si usano simultaneamente indicazioni di tempo diverse, le note che procedono parallelamente saranno poste nella stessa posizione sull’asse orizzontale. Tuttavia le stanghette dei vari rigli faranno sì che la spaziatura delle note sia meno regolare in ciascun rigo di quanto accadrebbe normalmente senza le diverse indicazioni di tempo.

## Divisione automatica delle note

Le note le cui durate eccedono il valore della battuta possono essere convertite automaticamente in note con legature di valore a cavallo delle stanghette sostituendo l’incisore `Note_heads_engraver` con `Completion_heads_engraver`. Analogamente, le pause le cui durate eccedono il valore della battuta possono essere divise automaticamente sostituendo `Rest_engraver` con `Completion_rest_engraver`. Nell’esempio seguente, le note e le pause che eccedono la durata di battuta vengono divise e le note sono anche connesse con legature di valore a cavallo della stanghetta.

```

\new Voice \with {
  \remove "Note_heads_engraver"
  \consists "Completion_heads_engraver"
  \remove "Rest_engraver"
  \consists "Completion_rest_engraver"
}
\relative {

```



```

c'2. c8 d4 e f g a b c8 c2 b4 a g16 f4 e d c8. c2 r1*2
}

```



Questi incisori dividono tutte le note e le pause in corrispondenza della stanghetta e inseriscono le legature di valore. Uno dei suoi usi possibili è la verifica di partiture complesse: se le misure non sono riempite interamente, le legature di valore mostrano esattamente di quanto è ecceduta ogni misura.

La proprietà `completionUnit` imposta la durata preferita per le note divise.

```
\new Voice \with {
  \remove "Note_heads_engraver"
  \consists "Completion_heads_engraver"
} \relative {
  \time 9/8 g\breve. d''4. \bar "||"
  \set completionUnit = #(ly:make-moment 3 8)
  g\breve. d4.
}
```



Questi incisori dividono le note che hanno una durata ridimensionata, come quelle dei gruppi irregolari, in note con lo stesso fattore di ridimensionamento della nota di input.

```
\new Voice \with {
  \remove "Note_heads_engraver"
  \consists "Completion_heads_engraver"
} \relative {
  \time 2/4 r4
  \tuplet 3/2 {g'4 a b}
  \scaleDurations 2/3 {g a b}
  g4*2/3 a b
  \tuplet 3/2 {g4 a b}
  r4
}
```



## Vedi anche

Glossario musicale: Sezione “legatura di valore” in *Glossario Musicale*

Manuale di apprendimento: Sezione “Gli incisori” in *Manuale di Apprendimento*, Sezione “Aggiungere e togliere gli incisori” in *Manuale di Apprendimento*.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Note\_heads\_engraver” in *Guida al Funzionamento Interno*, Sezione “Completion\_heads\_engraver” in *Guida al Funzionamento Interno*, Sezione “Rest\_engraver” in *Guida al Funzionamento Interno*, Sezione “Completion\_rest\_engraver” in *Guida al Funzionamento Interno*, Sezione “Forbid\_line\_break\_engraver” in *Guida al Funzionamento Interno*.

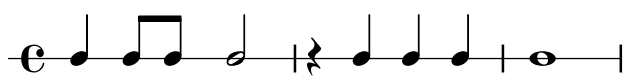
## Problemi noti e avvertimenti

In accordo col comportamento precedente, note e pause la cui durata è più lunga di una misura, come `c1*2`, sono divise in note prive di fattore di ridimensionamento, `{ c1 c1 }`. La proprietà `completionFactor` controlla questo comportamento e impostandola su `#f` fa sì che le note e le pause divise abbiano il fattore di ridimensionamento delle durate di input.

## Mostrare i ritmi della melodia

È possibile mostrare soltanto il ritmo di una melodia usando il rigo ritmico. Tutte le altezze delle note su tale rigo sono appiattite e il rigo stesso ha una sola linea

```
<<
  \new RhythmicStaff {
    \new Voice = "myRhythm" \relative {
      \time 4/4
      c'4 e8 f g2
      r4 g g f
      g1
    }
  }
  \new Lyrics {
    \lyricsto "myRhythm" {
      This is my song
      I like to sing
    }
  }
>>
```



This is my song      I like to    sing

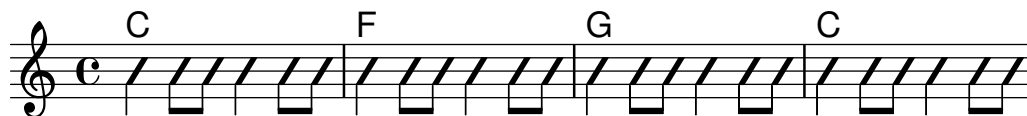
I diagrammi degli accordi per chitarra di solito mostrano i ritmi di accompagnamento. Si possono visualizzare usando l'incisore `Pitch_squash_engraver` e il comando `\improvisationOn`.

```
<<
  \new ChordNames {
    \chordmode {
      c1 f g c
    }
  }
  \new Voice \with {
    \consists "Pitch_squash_engraver"
  } \relative c'' {
    \improvisationOn
    c4 c8 c c4 c8 c
    f4 f8 f f4 f8 f
    g4 g8 g g4 g8 g
  }
>>
```

```

    c4 c8 c c4 c8 c
  }
>>

```



## Comandi predefiniti

`\improvisationOn`, `\improvisationOff`.

## Frammenti di codice selezionati

*Ritmi di accompagnamento per chitarra*

Per la musica per chitarra, è possibile mostrare i ritmi di accompagnamento, insieme alle note della melodia e ai nomi e ai diagrammi degli accordi.

```
\include "predefined-guitar-fretboards.ly"
```

```

<<
  \new ChordNames {
    \chordmode {
      c1 | f | g | c
    }
  }
  \new FretBoards {
    \chordmode {
      c1 | f | g | c
    }
  }
  \new Voice \with {
    \consists "Pitch_squash_engraver"
  } {
    \relative c'' {
      \improvisationOn
      c4 c8 c c4 c8 c
      f4 f8 f f4 f8 f
      g4 g8 g g4 g8 g
      c4 c8 c c4 c8 c
    }
  }
  \new Voice = "melody" {
    \relative c'' {
      c2 e4 e4
      f2. r4
      g2. a4
      e4 c2.
    }
  }
  \new Lyrics {
    \lyricsto "melody" {
      This is my song.
      I like to sing.
    }
  }

```

}  
 }  
 >>

C F G  
 x o o 3 2 1 1 3 4 2 1 1 2 1 3  
 This is my song. I like  
 4  
 C  
 x o o 3 2 1  
 to sing.

## Vedi anche

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “RhythmicStaff” in *Guida al Funzionamento Interno*, Sezione “Pitch\_squash\_engraver” in *Guida al Funzionamento Interno*.

### 1.2.4 Travature

#### Travature automatiche

Le travature sono inserite automaticamente:

```

\relative c' {
  \time 2/4 c8 c c c
  \time 6/8 c8 c c c8. c16 c8
}

```



Se queste impostazioni automatiche non sono soddisfacenti, si può definire esplicitamente la disposizione delle travature, come è spiegato in [\[Manual beams\]](#), pagina [\[undefined\]](#). Le travature *devono* essere inserite manualmente se devono estendersi oltre le pause.

La disposizione automatica delle travature, se non necessaria, può essere disabilitata con `\autoBeamOff` e riabilitata con `\autoBeamOn`:

```

\relative c' {

```

```

c4 c8 c8. c16 c8. c16 c8
\autoBeamOff
c4 c8 c8. c16 c8.
\autoBeamOn
c16 c8
}

```



**Nota:** Se si usano le travature per indicare i melismi nelle parti vocali, occorre disabilitare la disposizione automatica delle travature con `\autoBeamOff` e le travature devono essere indicate manualmente. L'uso di `\partcombine` insieme a `\autoBeamOff` può produrre risultati imprevisti. Si vedano i frammenti di codice per avere maggiori informazioni.

Si possono creare dei modelli di disposizione delle travature diversi da quelli automatici predefiniti, come è spiegato in [\[Setting automatic beam behavior\]](#), pagina [\[undefined\]](#).

## Comandi predefiniti

`\autoBeamOff`, `\autoBeamOn`.

## Frammenti di codice selezionati

*Travature che attraversano le interruzioni di linea*

Le interruzioni di linea sono di norma proibite quando le travature attraversano la stanghetta di una battuta. Si può cambiare questo comportamento nel modo seguente:

```

\relative c'' {
  \override Beam.breakable = ##t
  c8 c[ c] c[ c] c[ c] c[ \break
  c8] c[ c] c[ c] c[ c] c
}

```



*Modificare la distanza delle travature angolari*

Le travature angolari vengono inserite automaticamente quando viene rilevata un'ampia distanza tra le teste di nota. Questo comportamento può essere regolato attraverso la proprietà `auto-knee-gap`. Viene disegnata una travatura angolare se la distanza è più grande del valore di `auto-knee-gap` più la larghezza della travatura (che dipende dalla durata delle note e dall'inclinazione della travatura). Il valore predefinito di `auto-knee-gap` è 5.5 spazi rigo.

```
{
```

```
f8 f''8 f8 f''8
\override Beam.auto-knee-gap = #6
f8 f''8 f8 f''8
}
```



### *Partcombine e autoBeamOff*

La funzione `\autoBeamOff`, se usata insieme a `\partcombine`, può essere difficile da comprendere.

È preferibile usare invece

```
\set Staff.autoBeaming = ##f
```

per assicurarsi che la disposizione delle travature sia disabilitata per tutto il rigo.

`\partcombine` funziona con 3 voci – gambo in su singolo, gambo in giù singolo, gambo in su unito.

L'uso di `\autoBeamOff` all'interno del primo argomento di `partcombine` ha effetto sulla voce che è attiva al momento in cui la funzione viene elaborata, ovvero sul gambo in su singolo o sul gambo in giù unito. L'uso di `\autoBeamOff` nel secondo argomento avrà effetto sulla voce che ha il gambo in giù singolo.

Per poter usare `\autoBeamOff` per impedire tutte le disposizioni automatiche delle travature, se usato con `\partcombine`, è necessario richiamare tre volte la funzione `\autoBeamOff`.

```
{
  %\set Staff.autoBeaming = ##f % turns off all autobeaming
  \partcombine
  {
    \autoBeamOff % applies to split up stems
    \repeat unfold 4 a'16
    %\autoBeamOff % applies to combined up stems
    \repeat unfold 4 a'8
    \repeat unfold 4 a'16
  }
  {
    \autoBeamOff % applies to down stems
    \repeat unfold 4 f'8
    \repeat unfold 8 f'16 |
  }
}
```



Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Auto\_beam\_engraver” in *Guida al Funzionamento Interno*, Sezione “Beam\_engraver” in *Guida al Funzionamento Interno*, Sezione “Beam” in *Guida al Funzionamento Interno*, Sezione “BeamEvent” in *Guida al Funzionamento Interno*, Sezione “BeamForbidEvent” in *Guida al Funzionamento Interno*, Sezione “beam-interface” in *Guida al Funzionamento Interno*, Sezione “unbreakable-spanner-interface” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Le proprietà di una travatura sono determinate all’inizio della sua costruzione e qualsiasi ulteriore modifica alle sue proprietà che venga fatta prima che la travatura sia stata completata non avrà effetto finché non inizia la *successiva*, nuova travatura.

## Impostare il comportamento delle travature automatiche

Quando la disposizione automatica delle travature è abilitata, la disposizione delle travature è determinata da tre proprietà di contesto: `baseMoment`, `beatStructure` e `beamExceptions`. I valori predefiniti di queste variabili possono essere sovrascritti, come vedremo tra breve, oppure si possono anche cambiare i valori predefiniti stessi, come è spiegato in `[Time signature]`, pagina `[undefined]`.

Se è definita una regola `beamExceptions` per l’indicazione di tempo corrente, tale regola soltanto determina la disposizione delle travature; i valori di `baseMoment` e `beatStructure` vengono ignorati. Se non è definita alcuna regola `beamExceptions` per l’indicazione di tempo corrente, la disposizione delle travature è determinata dai valori di `baseMoment` e `beatStructure`.

## Disposizione delle travature basata su `baseMoment` e `beatStructure`

Dato che le indicazioni di tempo più comuni hanno delle regole `beamExceptions` già definite, occorre disabilitarle se la disposizione automatica deve basarsi su `baseMoment` e `beatStructure`. Le regole `beamExceptions` si disabilitano con questo comando

```
\set Timing.beamExceptions = #'()
```

Quando `beamExceptions` è impostato su `#'()`, o per impostazione esplicita o perché non sono state definite internamente le `beamExceptions` per l’indicazione di tempo corrente, le estremità delle travature si trovano sulle suddivisioni come specificato dalle proprietà di contesto `baseMoment` e `beatStructure`. `beatStructure` è una lista *scheme* che definisce la lunghezza di ogni suddivisione in rapporto alla misura in unità di `baseMoment`. Per impostazione predefinita, `baseMoment` è uno fratto il denominatore dell’indicazione di tempo e ogni unità di `baseMoment` corrisponde a una singola suddivisione.

Per ogni indicazione di tempo esistono valori separati per `beatStructure` e `baseMoment`. Le modifiche di queste variabili hanno effetto solo sulle indicazioni di tempo attive, dunque tali modifiche devono essere poste dopo il comando `\time` che inizia una nuova indicazione di tempo, non prima. I nuovi valori assegnati a una certa indicazione di tempo sono mantenuti e reintrodotti ogni volta che quell’indicazione di tempo viene ristabilita.

```
\relative c''{
  \time 5/16
  c16^"predefinito" c c c c |
  % È improbabile che per un tempo di 5/16 sia stata definita beamExceptions,
  % ma disabilitiamola lo stesso per sicurezza
  \set Timing.beamExceptions = #'()
  \set Timing.beatStructure = #'(2 3)
  c16^(2+3)" c c c c |
  \set Timing.beatStructure = #'(3 2)
```

```
c16^(3+2)" c c c c |
}
```



```
\relative {
  \time 4/4
  a'8^"predefinito" a a a a a a
  % Disabilita beamExceptions perché è senz'altro definita
  % per il tempo 4/4
  \set Timing.beamExceptions = #'()
  \set Timing.baseMoment = #(ly:make-moment 1/4)
  \set Timing.beatStructure = #'(1 1 1 1)
  a8^"cambiato" a a a a a a
}
```



Le modifiche alle impostazioni delle travature possono essere limitate a contesti specifici. Se non si specifica alcuna impostazione in un contesto di livello più basso, verrà applicata l'impostazione del contesto che lo contiene.

```
\new Staff {
  \time 7/8
  % Nessun bisogno di disabilitare beamExceptions perché non è definita per il tempo 7/8

  \set Staff.beatStructure = #'(2 3 2)
  <<
    \new Voice = one {
      \relative {
        a'8 a a a a a a
      }
    }
    \new Voice = two {
      \relative {
        \voiceTwo
        \set Voice.beatStructure = #'(1 3 3)
        f'8 f f f f f f
      }
    }
  >>
}
```





Quando si usano più voci, occorre specificare il contesto `Staff` se si vuole applicare la disposizione delle travature a tutte le voci del rigo:

```
\time 7/8
% ritmo 3-1-1-2
% Se non si specifica il contesto, la modifica viene applicata a Voice e quindi non fun
% Dato che le voci sono autogenerate, tutto il ritmo avrà come baseMoment (1 . 8)
\set beatStructure = #'(3 1 1 2)
<< \relative {a'8 a a a16 a a a a8 a} \\ \relative {f'4. f8 f f f} >>

% Funziona correttamente se si specifica il contesto Staff
\set Staff.beatStructure = #'(3 1 1 2)
<< \relative {a'8 a a a16 a a a a8 a} \\ \relative {f'4. f8 f f f} >>
```



Il valore di `baseMoment` può essere regolato in modo da cambiare il comportamento delle travature, se si vuole. In questo caso occorre cambiare anche il valore di `beatStructure` così che sia compatibile col nuovo valore di `baseMoment`.

```
\time 5/8
% Nessun bisogno di disabilitare beamExceptions perché non è definita per il tempo 5/8
\set Timing.baseMoment = #(ly:make-moment 1/16)
\set Timing.beatStructure = #'(7 3)
\repeat unfold 10 { a16 }
```



`baseMoment` è un *momento*, ovvero un'unità della durata musicale. Una quantità di tipo *moment* viene creata dalla funzione `scheme ly:make-moment`. Per maggiori informazioni su questa funzione, si veda [\[Time administration\]](#), pagina [\[Time administration\]](#).

Per impostazione predefinita, `baseMoment` ha un valore di uno fratto il denominatore dell'indicazione di tempo. Le eccezioni a questa regola si trovano in `scm/time-signature-settings.scm`.

### ***Disposizione delle travature con beamExceptions***

Le regole speciali di disposizione automatica delle travature (diverse da quelle che determinano la corrispondenza della travatura alla suddivisione) sono definite nella proprietà `beamExceptions`.

Il valore di `beamExceptions`, una struttura dati Scheme piuttosto complessa, è più facile da generare con la funzione `\beamExceptions`. A tale funzione viene passato uno o più schemi ritmici della misura, specificati con travature manuali. Le misure devono essere separate da un controllo di battuta | dato che la funzione non ha altro modo per determinare la lunghezza della misura. Ecco un semplice esempio:

```
\relative c'' {
  \time 3/16
  \set Timing.beatStructure = #'(2 1)
  \set Timing.beamExceptions =
    \beamExceptions { 32[ 32] 32[ 32] 32[ 32] }
```

```
c16 c c |
\repeat unfold 6 { c32 } |
}
```



**Nota:** Il valore di `beamExceptions` deve essere una lista *completa* di eccezioni, ovvero bisogna includere tutte le eccezioni che si vogliono applicare. Non è possibile aggiungere, rimuovere o modificare soltanto una eccezione. Anche se questo può sembrare scomodo, significa anche che non c'è bisogno di conoscere le attuali impostazioni delle travature per poter specificare un nuovo modello di disposizione delle travature.

Quando cambia l'indicazione di tempo, vengono impostati i valori predefiniti di `Timing.baseMoment`, `Timing.beatStructure` e `Timing.beamExceptions`. L'impostazione dell'indicazione di tempo ripristina le impostazioni automatiche delle travature del contesto `Timing` ai valori predefiniti.

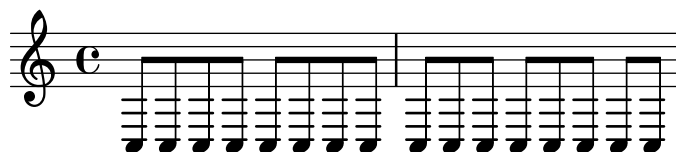
```
\relative a' {
  \time 6/8
  \repeat unfold 6 { a8 }
  % raggruppamento (4 + 2)
  \set Timing.beatStructure = #'(4 2)
  \repeat unfold 6 { a8 }
  % ritorno al comportamento predefinito
  \time 6/8
  \repeat unfold 6 { a8 }
}
```



Le impostazioni predefinite della disposizione automatica delle travature per ogni tempo sono definite in `scm/time-signature-settings.scm`. La loro modifica è descritta in `<undefined> [Time signature]`, pagina `<undefined>`.

Molte impostazioni di travature automatiche per le indicazioni di tempo hanno un elemento `beamExceptions`. Ad esempio, il tempo 4/4 cerca di creare due travature nella misura se ci sono solo note di un ottavo. La regola `beamExceptions` può sovrascrivere l'impostazione di `beatStructure` se `beamExceptions` non viene annullato.

```
\time 4/4
\set Timing.baseMoment = #(ly:make-moment 1/8)
\set Timing.beatStructure = #'(3 3 2)
% Le travature non saranno raggruppate in (3 3 2) a causa di beamExceptions
\repeat unfold 8 {c8} |
% Il raggruppamento delle travature è (3 3 2) perché abbiamo tolto le impostazioni predefinite
\set Timing.beamExceptions = #'()
\repeat unfold 8 {c8}
```



Analogamente, le note di un ottavo in un tempo 3/4 sono raggruppate in un'unica travatura. Per raggrupparle secondo le suddivisioni, azzera `beamExceptions`.

```
\time 3/4
% il comportamento predefinito è un gruppo di (6) a causa di beamExceptions
\repeat unfold 6 {a8} |
% Le travature saranno raggruppate in (1 1 1) a causa dei valori predefiniti di baseMoment
\set Timing.beamExceptions = #'()
\repeat unfold 6 {a8}
```



Spesso, nelle partiture di età classica e romantica, le travature iniziano a metà della misura in un tempo 3/4; ma la pratica moderna preferisce evitare l'impressione ingannevole di un tempo 6/8 (vedi Gould, p. 153). Situazioni simili si incontrano anche per il tempo 3/8. Questo comportamento è controllato dalla proprietà di contesto `beamHalfMeasure`, che ha effetto soltanto sulle indicazioni di tempo che hanno 3 come numeratore:

```
\relative a' {
  \time 3/4
  r4. a8 a a |
  \set Timing.beamHalfMeasure = ##f
  r4. a8 a a |
}
```



### *Come funziona la disposizione automatica delle travature*

Quando la disposizione automatica delle travature è abilitata, la disposizione delle travature è determinata dalle proprietà di contesto `baseMoment`, `beatStructure` e `beamExceptions`.

Nel determinare l'aspetto delle travature vengono applicate le seguenti regole, in ordine di priorità:

- Se si specifica una travatura manuale con [...] imposta la travatura in quel modo, altrimenti
- se è definita una regola di fine della travatura in `beamExceptions` per il tipo di travatura in questione, la usa per determinare i punti corretti in cui le travature possono terminare, altrimenti
- se è definita una regola di fine della travatura in `beamExceptions` per un tipo di travatura più lunga, la usa per determinare i punti corretti in cui le travature possono terminare, altrimenti
- usa i valori di `baseMoment` e `beatStructure` per determinare l'estensione delle suddivisioni della misura e terminare le travature alla fine delle suddivisioni.

Nelle regole precedenti, il *tipo di travatura* è la durata della nota più corta nel gruppo della travatura.

Le regole predefinite per le travature si trovano in `scm/time-signature-settings.scm`.

## Frammenti di codice selezionati

### *Suddividere le travature*

Le travature di note consecutive di un sedicesimo (o più brevi) non vengono suddivise, ovvero i tre (o più) tratti della travatura si estendono, senza spezzarsi, sugli interi gruppi di note. Questo comportamento può essere modificato in modo da suddividere le travature in sottogruppi attraverso la proprietà `subdivideBeams`. Se impostata, le travature che comprendono più sottogruppi verranno suddivise a intervalli definiti dal valore attuale di `baseMoment`, riducendo le travature multiple al numero di travature che indica la lunghezza del sottogruppo. Si noti che `baseMoment`, se non impostata esplicitamente, equivale a uno fratto il denominatore dell'attuale indicazione di tempo. Deve quindi essere impostata su una frazione che stabilisca la durata del sottogruppo di travature; lo si può fare usando la funzione `ly:make-moment`, come è mostrato in questo frammento di codice. Inoltre quando `baseMoment` cambia, anche `beatStructure` deve essere modificato per accordarsi con `baseMoment`:

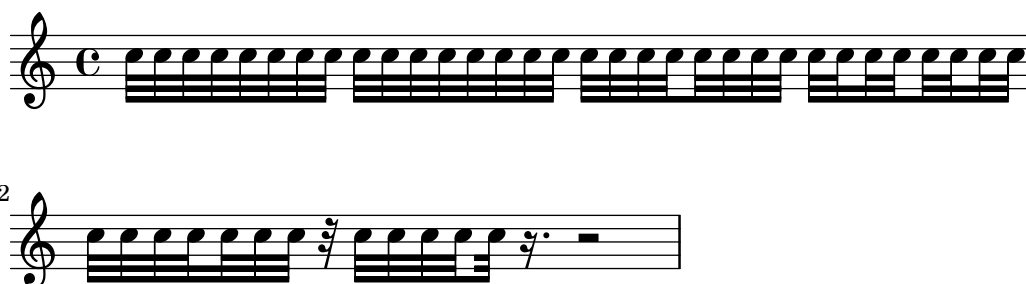
```
\relative c'' {
  c32[ c c c c c c c]
  \set subdivideBeams = ##t
  c32[ c c c c c c c]

  % Set beam sub-group length to an eighth note
  \set baseMoment = #(ly:make-moment 1/8)
  \set beatStructure = #'(2 2 2 2)
  c32[ c c c c c c c]

  % Set beam sub-group length to a sixteenth note
  \set baseMoment = #(ly:make-moment 1/16)
  \set beatStructure = #'(4 4 4 4)
  c32[ c c c c c c c]

  % Shorten beam by 1/32
  \set baseMoment = #(ly:make-moment 1/8)
  \set beatStructure = #'(2 2 2 2)
  c32[ c c c c c c] r32

  % Shorten beam by 3/32
  \set baseMoment = #(ly:make-moment 1/8)
  \set beatStructure = #'(2 2 2 2)
  c32[ c c c c] r16.
  r2
}
```



*Travatura che segue strettamente il battito*

Si possono impostare i tratti di suddivisione della travatura in modo che siano rivolti verso la relativa pulsazione. La prima travatura fa sì che non spuntino i tratti di suddivisione (comportamento predefinito); la seconda travatura è orientata verso la pulsazione.

```
\relative c'' {
  \time 6/8
  a8. a16 a a
  \set strictBeatBeaming = ##t
  a8. a16 a a
}
```



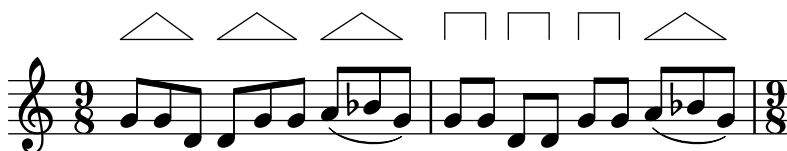
*Segni per la conduzione, segni di raggruppamento della misura*

Il raggruppamento delle pulsazioni all'interno della misura è regolato dalla proprietà di contesto `beatStructure`. I valori di `beatStructure` per varie indicazioni di tempo vengono stabiliti in `scm/time-signature-settings.scm`. Questi valori possono essere impostati o modificati con `\set`. Altrimenti, si può usare `\time` per impostare sia l'indicazione di tempo che la struttura delle pulsazioni. Per farlo si specifica il raggruppamento interno delle pulsazioni in una misura in una lista di numeri (nella sintassi di Scheme) prima dell'indicazione di tempo.

`\time` agisce nel contesto `Timing`, dunque non reimposterà i i valori di `beatStructure` e `baseMoment` che sono impostati in altri contesti di più basso livello, come `Voice`.

Se si include l'incisore `Measure_grouping_engraver` in uno dei contesti che regolano l'aspetto, appariranno i segni di raggruppamento della misura. Tali segni facilitano la lettura di musica moderna ritmicamente complessa. Nell'esempio la misura di 9/8 è raggruppata in due diversi schemi usando due metodi differenti, mentre la misura di 5/8 è raggruppata in base alle impostazioni predefinite in `scm/time-signature-settings.scm`:

```
\score {
  \new Voice \relative c'' {
    \time 9/8
    g8 g d d g g a( bes g) |
    \set Timing.beatStructure = #'(2 2 2 3)
    g8 g d d g g a( bes g) |
    \time #'(4 5) 9/8
    g8 g d d g g a( bes g) |
    \time 5/8
    a4. g4 |
  }
  \layout {
    \context {
      \Staff
      \consists "Measure_grouping_engraver"
    }
  }
}
```

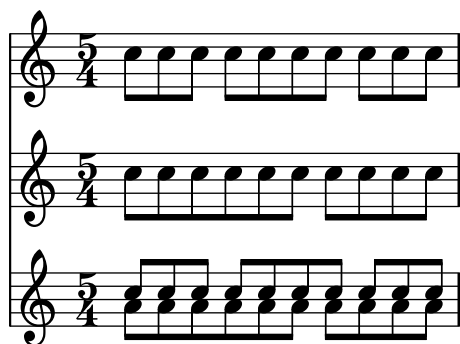




### *Estremità delle travature nel contesto Score*

Le regole relative alle estremità delle travature definite nel contesto **Score** si applicano a tutti i righe, ma possono essere modificate anche ai livelli **Staff** e **Voice**:

```
\relative c'' {
  \time 5/4
  % Set default beaming for all staves
  \set Score.baseMoment = #(ly:make-moment 1/8)
  \set Score.beatStructure = #'(3 4 3)
  <<
    \new Staff {
      c8 c c c c c c c c c
    }
    \new Staff {
      % Modify beaming for just this staff
      \set Staff.beatStructure = #'(6 4)
      c8 c c c c c c c c c
    }
    \new Staff {
      % Inherit beaming from Score context
      <<
        {
          \voiceOne
          c8 c c c c c c c c c
        }
        % Modify beaming for this voice only
        \new Voice {
          \voiceTwo
          \set Voice.beatStructure = #'(6 4)
          a8 a a a a a a a a a
        }
      >>
    }
  >>
}
```



### Vedi anche

Guida alla notazione: [\[Time signature\]](#), pagina [\[undefined\]](#).

File installati: `scm/time-signature-settings.scm`.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

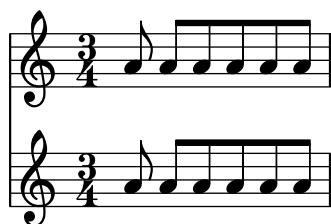
Guida al funzionamento interno: Sezione “Auto\_beam\_engraver” in *Guida al Funzionamento Interno*, Sezione “Beam” in *Guida al Funzionamento Interno*, Sezione “BeamForbidEvent” in *Guida al Funzionamento Interno*, Sezione “beam-interface” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Se una partitura finisce prima del termine di una travatura automatica, cui mancano ancora delle note, quest’ultima travatura non apparirà. Lo stesso vale per le voci polifoniche, inserite con `<< ... \ \ ... >>`. Una voce polifonica non apparirà se termina quando una travatura automatica è ancora in attesa di note. Per aggirare questi problemi occorre impostare manualmente l’ultima travatura della voce o della partitura.

**Timing** è un alias del contesto **Score**. Questo significa che la modifica della disposizione delle travature in un rigo avrà effetto anche sugli altri rigi. Quindi un’impostazione di tempo in un rigo successivo reimposterà la disposizione personalizzata delle travature definita in un rigo precedente. Per evitare questo problema si può impostare l’indicazione di tempo su un solo rigo.

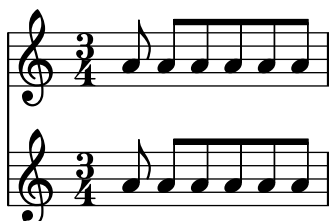
```
<<
  \new Staff {
    \time 3/4
    \set Timing.baseMoment = #(ly:make-moment 1/8)
    \set Timing.beatStructure = #'(1 5)
    \set Timing.beamExceptions = #'()
    \repeat unfold 6 { a'8 }
  }
  \new Staff {
    \repeat unfold 6 { a'8 }
  }
>>
```



Si possono cambiare anche le impostazioni predefinite delle travature, in modo che sia usata sempre la disposizione delle travature desiderata. Le modifiche nelle impostazioni della travatura automatica per le indicazioni di tempo sono descritte in `<undefined>` [Time signature], pagina `<undefined>`.

```
<<
  \new Staff {
    \overrideTimeSignatureSettings
      3/4                % timeSignatureFraction
      1/8                % baseMomentFraction
      #'(1 5)           % beatStructure
      #'()              % beamExceptions
    \time 3/4
    \repeat unfold 6 { a'8 }
  }
>>
```

```
\new Staff {
  \time 3/4
  \repeat unfold 6 { a'8 }
}
>>
```



## Travature manuali

In alcuni casi potrebbe essere necessario scavalcare l'algoritmo di disposizione automatica delle travature. Ad esempio, questo algoritmo non inserirà delle travature tra le pause o tra le stanghette; e nelle partiture corali la disposizione delle travature è spesso determinato dall'articolazione del testo piuttosto che da quella musicale. Tali travature possono essere specificate manualmente indicandone l'inizio e la fine con [ e ].

```
\relative { r4 r8[ g' a r] r g[ | a] r }
```



La direzione delle travature può essere impostata manualmente attraverso gli indicatori di direzione:

```
\relative { c''8^[ d e] c,_[ d e f g] }
```



Le note individuali possono essere contrassegnate con `\noBeam` per impedire che vengano inserite in una travatura:

```
\relative {
  \time 2/4
  c''8 c\noBeam c c
}
```

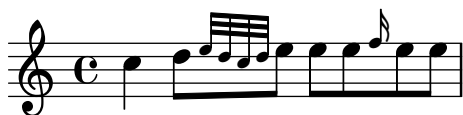


Le travature degli abbellimenti e quelle delle note normali possono coesistere simultaneamente. Gli abbellimenti privi di travatura non vengono inseriti nella travatura delle note normali.

```
\relative {
  c''4 d8[
  \grace { e32 d c d }
]
```



```
e8] e[ e
\grace { f16 }
e8 e]
}
```



Si può ottenere un controllo manuale delle travature ancora più preciso agendo sulle proprietà `stemLeftBeamCount` e `stemRightBeamCount`, che specificano il numero di travature da creare a sinistra e a destra della nota successiva. Se una di queste proprietà viene impostata, il suo valore verrà usato una volta sola, e la proprietà sarà poi cancellata. In questo esempio, l'ultima nota `f` ha una sola travatura a sinistra: la travatura corrispondente alla sottodivisione di un ottavo all'interno dell'intero raggruppamento.

```
\relative a' {
  a8[ r16 f g a]
  a8[ r16
  \set stemLeftBeamCount = #2
  \set stemRightBeamCount = #1
  f16
  \set stemLeftBeamCount = #1
  g16 a]
}
```



## Comandi predefiniti

`\noBeam.`

## Frammenti di codice selezionati

### *Code e punte delle travature*

È possibile ottenere delle codette su note isolate e dei tratti di suddivisione all'estremità della travatura con una combinazione di `stemLeftBeamCount`, `stemRightBeamCount` e una coppia di indicatori della travatura `[]`.

Per ottenere delle codette rivolte a destra, si usa la coppia di indicatori `[]` e si imposta `stemLeftBeamCount` a zero (vedi Example 1).

Per ottenere delle codette rivolte a sinistra, si imposta invece `stemRightBeamCount` (Example 2).

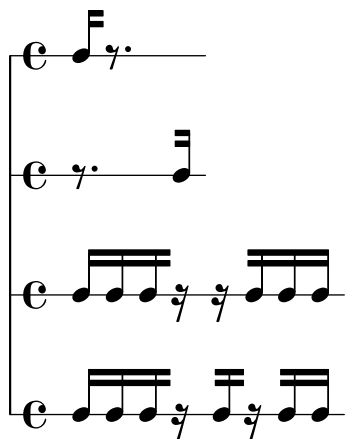
Perché i tratti di suddivisione alla fine di un gruppo di note unite da travatura siano rivolti a destra, si imposta `stemRightBeamCount` su un valore positivo. Perché i tratti di suddivisione all'inizio di un gruppo di note unite da travatura siano rivolti a sinistra, si imposta invece `stemLeftBeamCount` (Example 3).

Talvolta, ad esempio per una nota isolata circondata da pause, ha senso avere una coda che punti sia a destra che a sinistra. Lo si può fare con una coppia di indicatori di travatura `[]` da soli (Example 4).

(Nota che `\set stemLeftBeamCount` è sempre equivalente a `\once \set`. In altre parole, le impostazioni che definiscono il conteggio delle travature non “permangono”, quindi la coppia di

code attaccate al 16[] solitario nell'ultimo esempio non hanno nulla a che fare con l'impostazione \set di due note prima.)

```
\score {
  <<
    % Example 1
    \new RhythmicStaff {
      \set stemLeftBeamCount = #0
      c16[]
      r8.
    }
    % Example 2
    \new RhythmicStaff {
      r8.
      \set stemRightBeamCount = #0
      16[]
    }
    % Example 3
    \new RhythmicStaff {
      16 16
      \set stemRightBeamCount = #2
      16 r r
      \set stemLeftBeamCount = #2
      16 16 16
    }
    % Example 4
    \new RhythmicStaff {
      16 16
      \set stemRightBeamCount = #2
      16 r16
      16[]
      r16
      \set stemLeftBeamCount = #2
      16 16
    }
  >>
}
```



## Vedi anche

Guida alla notazione: Sezione 5.4.2 [Direction and placement], pagina 611, [\[Grace notes\]](#), pagina [\(undefined\)](#).

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Beam” in *Guida al Funzionamento Interno*, Sezione “BeamEvent” in *Guida al Funzionamento Interno*, Sezione “Beam\_engraver” in *Guida al Funzionamento Interno*, Sezione “beam-interface” in *Guida al Funzionamento Interno*, Sezione “Stem\_engraver” in *Guida al Funzionamento Interno*.

## Travature a raggiera

Le travature a raggiera servono a indicare che un gruppo di note determinato deve essere eseguito a un tempo progressivamente accelerato (o rallentato), senza cambiare l’andamento complessivo del brano. L’estensione della travatura a raggiera deve essere indicato a mano con [ e ], e la convergenza o divergenza delle travature si determina specificando la la direzione della proprietà Beam di `grow-direction`.

Perché il *ritardando* o l’*accelerando* indicati dalla travatura a raggiera trovino riscontro nella disposizione delle note e nell’esecuzione del file MIDI, le note devono essere raggruppate in un’espressione musicale delimitata da parentesi graffe e preceduta dal comando `\featherDurations`, che specifica il rapporto tra le durate delle prime e delle ultime note del gruppo.

Le parentesi quadre indicano l’estensione della travatura, mentre quelle graffe indicano quali note devono avere una durata modificata. Di norma queste parentesi delimitano lo stesso gruppo di note, ma questo non è tassativo: i due comandi sono indipendenti.

Nell’esempio seguente le otto note da un sedicesimo occupano esattamente lo stesso tempo di una nota di due quarti, ma la prima nota dura la metà dell’ultima e le note intermedie si allungano gradualmente. Le prime quattro note da un trentaduesimo sono progressivamente più veloci, mentre le ultime quattro presentano lo stesso tempo.

```
\relative c' {
  \override Beam.grow-direction = #LEFT
  \featherDurations #(ly:make-moment 2/1)
  { c16[ c c c c c c c c] }
  \override Beam.grow-direction = #RIGHT
  \featherDurations #(ly:make-moment 2/3)
  { c32[ d e f] }
  % ripristina le travature normali
  \override Beam.grow-direction = #'()
  { g32[ a b c] }
}
```



La spaziatura rappresenta la durata effettiva delle note solo in modo approssimato, mentre il tempo nel file MIDI è esatto.

## Comandi predefiniti

`\featherDurations`.

## Vedi anche

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

## Problemi noti e avvertimenti

Il comando `\featherDurations` funziona solamente con frammenti di musica molto brevi e quando i numeri della frazione sono piccoli.

### 1.2.5 Battute

#### Stanghette

Le stanghette delimitano le misure e sono usate anche per indicare i ritornelli. Di norma, le stanghette semplici sono inserite automaticamente in base all'indicazione di tempo.

Si possono inserire altri tipi di stanghette col comando `\bar`. Ad esempio, di solito si usa una stanghetta finale al termine di un brano:

```
\relative { e'4 d c2 \bar "|." }
```



Se l'ultima nota di una misura non termina entro la stanghetta inserita automaticamente, non viene segnalato un errore: si presuppone che la nota continui nella misura successiva. Ma se ci sono tante misure simili in sequenza, la musica potrebbe apparire compressa oppure scorrere fuori dalla pagina. Questo accade perché le interruzioni di linea automatiche si verificano solo al termine di misure complete, ovvero quando tutte le note terminano prima dell'inizio di una misura.

**Nota:** Una durata errata può impedire un'interruzione di linea, causando una linea di musica altamente compressa oppure a musica che prosegue fuori dalla pagina.

Le interruzioni di linea sono permesse anche in caso si stanghette inserite a mano anche all'interno di misure incomplete. Per permettere un'interruzione di linea senza che appaia una stanghetta si usa:

```
\bar ""
```

Questo comando inserirà una stanghetta invisibile e consentirà (senza però forzarla) un'interruzione di linea in questo punto. Il conteggio dei numeri di battuta non incrementa. Per forzare un'interruzione di linea si veda `\bar` [Line breaking], pagina `\bar`.

Si possono inserire questa e altre stanghette speciali in qualsiasi punto. Quando coincidono con la fine di una misura, sostituiscono la stanghetta semplice che sarebbe stata posta automaticamente. Quando non coincidono con la fine di una misura, la stanghetta specificata viene inserita in quel punto.

Si noti che le stanghette manuali hanno una funzione puramente visiva. Non hanno alcun effetto sulle proprietà di una normale stanghetta, come i numeri della misura, le alterazioni, le interruzioni di linea, etc. Non influiscono nemmeno sul conteggio e sulla posizione delle stanghette automatiche successive. Quando una stanghetta manuale è posta nel punto in cui si trova già una normale stanghetta, le caratteristiche della stanghetta originale non sono alterate.

Sono disponibili per l'inserimento manuale due tipi di stanghette semplici e cinque tipi di stanghette doppie:

```
\relative {
  f'1 \bar "|"
  f1 \bar "."
  g1 \bar "||"
```

```

a1 \bar ".|"
b1 \bar ".."
c1 \bar "|.|"
d1 \bar "|."
e1
}

```

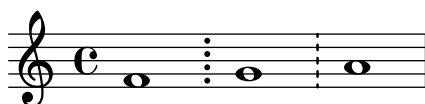


oltre alle stanghette puntate e tratteggiate:

```

\relative {
  f'1 \bar ";"
  g1 \bar "!"
  a1
}

```

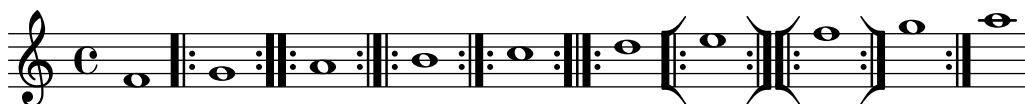


e a nove tipi di stanghette per le ripetizioni:

```

\relative {
  f'1 \bar ".|:"
  g1 \bar ":\.:"
  a1 \bar ":\.|"
  b1 \bar ":\.:"
  c1 \bar ":\.|"
  d1 \bar "[|:"
  e1 \bar ":\]|[:]"
  f1 \bar ":\]|"
  g1 \bar ":\.|"
  a1
}

```

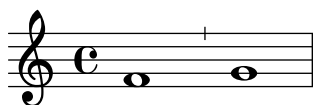


Inoltre, una stanghetta può apparire come un semplice segno di spunta:

```

f'1 \bar "'" g'1

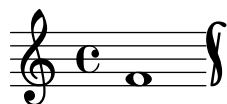
```



Tuttavia, dato che questi segni di spunta sono tipicamente usati nella notazione gregoriana, è preferibile usare `\divisioMinima`, come è descritto nella sezione [Divisiones], pagina 444, della parte dedicata al canto gregoriano.

Lilypond supporta la notazione gregoriana russa e fornisce una stanghetta speciale per questo tipo di notazione:

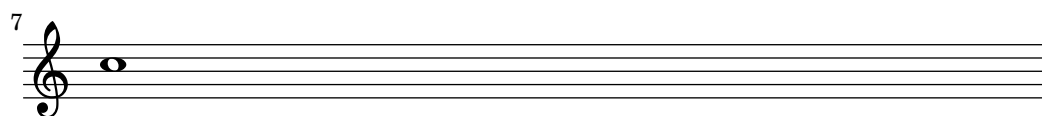
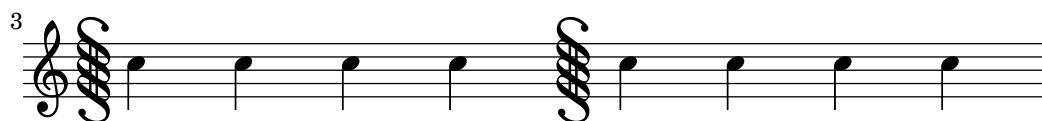
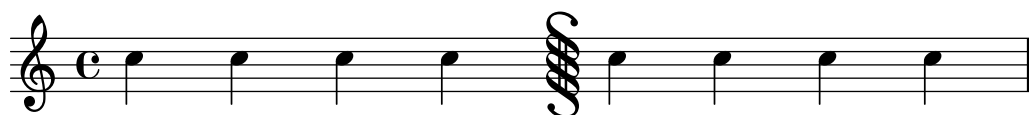
```
f'1 \bar "k"
```



I dettagli di questo tipo di notazione sono spiegati in Sezione 2.9.5 [Typesetting Kievan square notation], pagina 453.

Per i segni di tipo segno interni al rigo, ci sono tre tipi di stanghette che differiscono nel comportamento quando incontrano un'interruzione di linea:

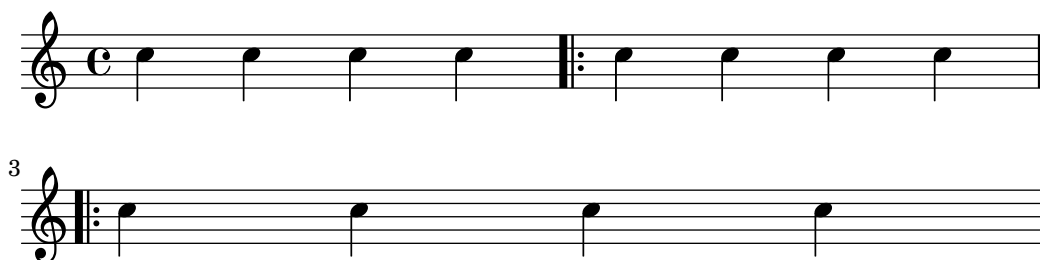
```
\relative c'' {
  c4 c c c
  \bar "S"
  c4 c c c \break
  \bar "S"
  c4 c c c
  \bar "S-|"
  c4 c c c \break
  \bar "S-|"
  c4 c c c
  \bar "S-S"
  c4 c c c \break
  \bar "S-S"
  c1
}
```



Sebbene LilyPond preveda l'inserimento manuale delle stanghette che indicano i ritornelli, ciò non consente il riconoscimento della musica come una sezione da ripetere. Tali sezioni devono essere inserite con i vari comandi di ripetizione (vedi [\[Repeats\]](#), pagina [\[Repeats\]](#)), che creano automaticamente le stanghette appropriate.

Inoltre si può specificare ".|:-||", che è equivalente a ".|:" tranne in presenza di un'interruzione di linea, dove crea una doppia stanghetta alla fine della linea e una stanghetta di inizio ripetizione all'inizio della linea successiva.

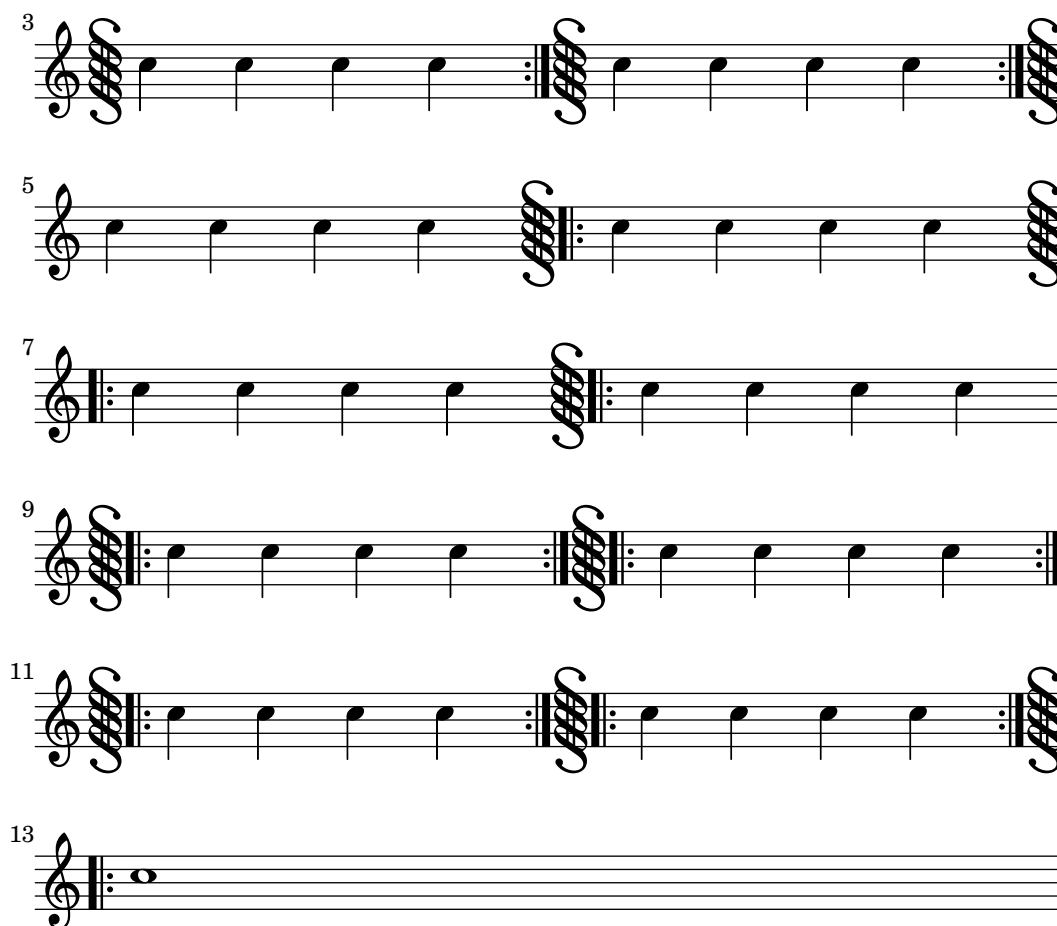
```
\relative c'' {
  c4 c c c
  \bar ".|:-||"
  c4 c c c \break
  \bar ".|:-||"
  c4 c c c
}
```



Esistono sei diverse combinazioni di ripetizioni e indicazioni di segno:

```
\relative c'' {
  c4 c c c
  \bar " :|.S"
  c4 c c c \break
  \bar " :|.S"
  c4 c c c
  \bar " :|.S-S"
  c4 c c c \break
  \bar " :|.S-S"
  c4 c c c
  \bar "S.|:-S"
  c4 c c c \break
  \bar "S.|:-S"
  c4 c c c
  \bar "S.|"
  c4 c c c \break
  \bar "S.|"
  c4 c c c
  \bar " :|.S.|"
  c4 c c c \break
  \bar " :|.S.|"
  c4 c c c
  \bar " :|.S.|-S"
  c4 c c c \break
  \bar " :|.S.|-S"
  c1
}
```





Esiste inoltre un comando `\inStaffSegno` che crea una stanghetta con segno in congiunzione con un'appropriata stanghetta di ripetizione se usata con un comando `\repeat volta`, vedi [\[Normal repeats\]](#), pagina [\[undefined\]](#)..

Si possono definire nuovi tipi di stanghetta con `\defineBarLine`:

```
\defineBarLine tipo-stanghetta #'(fine inizio span)
```

Le variabili di `\defineBarLine` possono includere la stringa 'vuota' "", che è equivalente a una stanghetta invisibile. Oppure possono essere impostate su `#f`, che fa sì che non appaia alcuna stanghetta.

Dopo averla definita, si può richiamare la nuova stanghetta col comando `\bar tipo-stanghetta`.

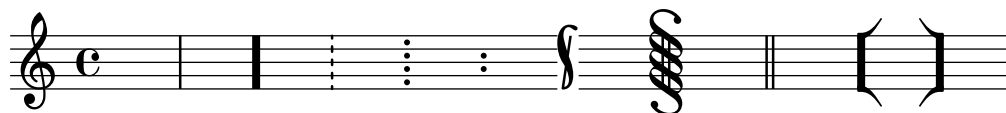
Attualmente sono disponibili dieci tipi di stanghetta:

```
\defineBarLine ":" #'(" ":" ")
\defineBarLine "=" #'("=" " ")
\defineBarLine "[" #'("[" " ")
\defineBarLine "]" #'("]" " ")
```

```
\new Staff {
  s1 \bar "|"
  s1 \bar "."
  s1 \bar "!"
  s1 \bar ";"
  s1 \bar ":"
  s1 \bar "k"
  s1 \bar "S"
  s1 \bar "="
  s1 \bar "["
  s1 \bar "]"
```



```
s1 \bar ""
}
```

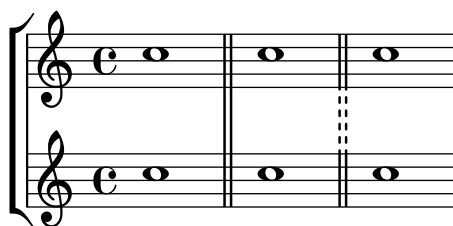


La stanghetta "=" crea una stanghetta doppia da combinare con il segno. Non va usata per creare una stanghetta doppia indipendente; in questo caso è preferibile usare `\bar "||"`.

Il segno "-" introduce le annotazioni alle stanghette che servono a distinguere quelle che hanno aspetto identico ma un diverso comportamento in corrispondenza delle interruzioni di linea e/o un diverso modo di connettere le stanghette tra i righi. La parte che segue il segno "-" non viene usato per costruire la stanghetta.

```
\defineBarLine "||-dashedSpan" #'("||" "" "!!")
```

```
\new StaffGroup <<
  \new Staff \relative c'' {
    c1 \bar "||"
    c1 \bar "||-dashedSpan"
    c1
  }
  \new Staff \relative c'' {
    c1
    c1
    c1
  }
>>
```

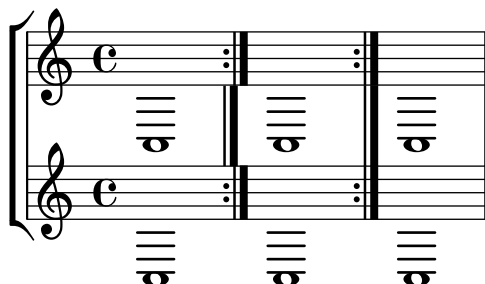


Inoltre, lo spazio " " fa da spaziatore e fa sì che le stanghette tra i righi siano allineate correttamente alle stanghette principali:

```
\defineBarLine ":-sbugliata" #'(":-|." "" " |.")
\defineBarLine ":-giusta" #'(":-|." "" " |.")
```

```
\new StaffGroup <<
  \new Staff {
    c1 \bar ":-sbugliata"
    c1 \bar ":-giusta"
    c1
  }
  \new Staff {
    c1
    c1
    c1
  }
>>
```

&gt;&gt;



Se servono ulteriori elementi, LilyPond fornisce un modo semplice per definirli. Maggiori informazioni sulla modifica e l'aggiunta delle stanghette sono presenti nel file `scm/bar-line.scm`.

Nelle partiture con molti righi, un comando `\bar` inserito in un rigo viene applicato automaticamente a tutti i righi. Le stanghette risultanti sono connesse tra i diversi righi di un `StaffGroup`, `PianoStaff` o `GrandStaff`.

&lt;&lt;

```
\new StaffGroup <<
  \new Staff \relative {
    e'4 d
    \bar "||"
    f4 e
  }
  \new Staff \relative { \clef bass c'4 g e g }
```

&gt;&gt;

```
\new Staff \relative { \clef bass c'2 c2 }
```

&gt;&gt;



Il comando `'\bar tipo-stanghetta'` è una scorciatoia di `'\set Timing.whichBar = tipo-stanghetta'`. Una stanghetta viene creata ogni volta che si imposta la proprietà `whichBar`.

Il tipo di stanghetta predefinita per le stanghette inserite automaticamente è `"|"`. Si può modificare in qualsiasi momento con `'\set Timing.defaultBarType = tipo-stanghetta'`.

## Vedi anche

Guida alla notazione: [\[Line breaking\]](#), pagina [\[Repeats\]](#), pagina [\[Grouping staves\]](#), pagina [\[Timing\]](#).

File installati: `scm/bar-line.scm`.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “BarLine” in *Guida al Funzionamento Interno* (creata al livello `Staff`), Sezione “SpanBar” in *Guida al Funzionamento Interno* (tra i righi), Sezione “Timing\_translator” in *Guida al Funzionamento Interno* (per le proprietà di Timing).

## Numeri di battuta

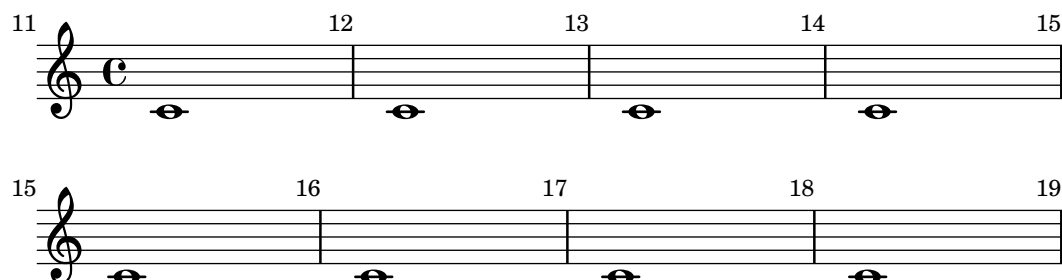
I numeri di battuta appaiono all'inizio di ogni linea tranne la prima. Il numero viene salvato nella proprietà `currentBarNumber`, che viene aggiornata automaticamente per ogni misura. Può anche essere impostata a mano:

```
\relative c' {
  c1 c c c
  \break
  \set Score.currentBarNumber = #50
  c1 c c c
}
```



I numeri di battuta possono essere mostrati a intervalli regolari anziché solo all'inizio di ogni linea. Per farlo occorre sovrascrivere il comportamento predefinito e permettere ai numeri di battuta di apparire anche in punti diversi dall'inizio della linea. Questo comportamento è regolato dalla proprietà `break-visibility` di `BarNumber`, che considera tre valori impostabili su `#t` o `#f`, i quali indicano se il numero di battuta corrispondente debba essere visibile o no. L'ordine dei tre valori è `end of line visible`, `middle of line visible`, `beginning of line visible`. Nell'esempio seguente i numeri di battuta compaiono in tutti i punti possibili:

```
\relative c' {
  \override Score.BarNumber.break-visibility = ##(#t #t #t)
  \set Score.currentBarNumber = #11
  % Permette la visualizzazione del primo numero di battuta
  \bar ""
  c1 | c | c | c
  \break
  c1 | c | c | c
}
```



## Frammenti di codice selezionati

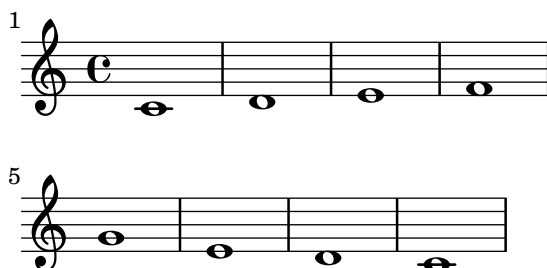
*Mostrare il numero di battuta nella prima misura*

Il primo numero di battuta di una partitura viene soppresso se è inferiore o uguale a '1'. Se si imposta `barNumberVisibility` su `all-bar-numbers-visible`, verrà mostrato il numero di

battuta della prima misura e di tutte quelle successive. Si noti che perché funzioni è necessario inserire una stanghetta invisibile prima della prima nota.

```
\layout {
  indent = 0
  ragged-right = ##t
}

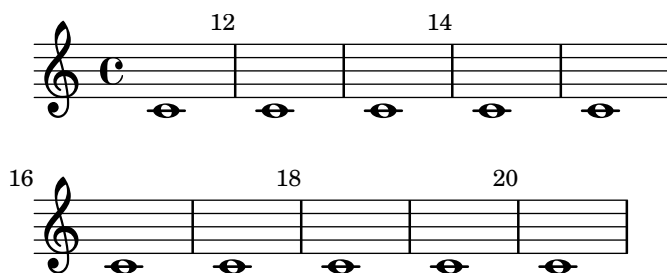
\relative c' {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \bar ""
  c1 | d | e | f \break
  g1 | e | d | c
}
```



*Mostrare i numeri di battuta a intervalli regolari*

I numeri di battuta possono essere resi visibili a intervalli regolari attraverso la proprietà `barNumberVisibility`. In questo esempio vengono mostrati ogni due misure eccetto alla fine della linea.

```
\relative c' {
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \set Score.currentBarNumber = #11
  % Permit first bar number to be printed
  \bar ""
  % Print a bar number every second measure
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
  c1 | c | c | c | c
  \break
  c1 | c | c | c | c
}
```



*Stampare i numeri di battuta a intervalli regolari variabili*

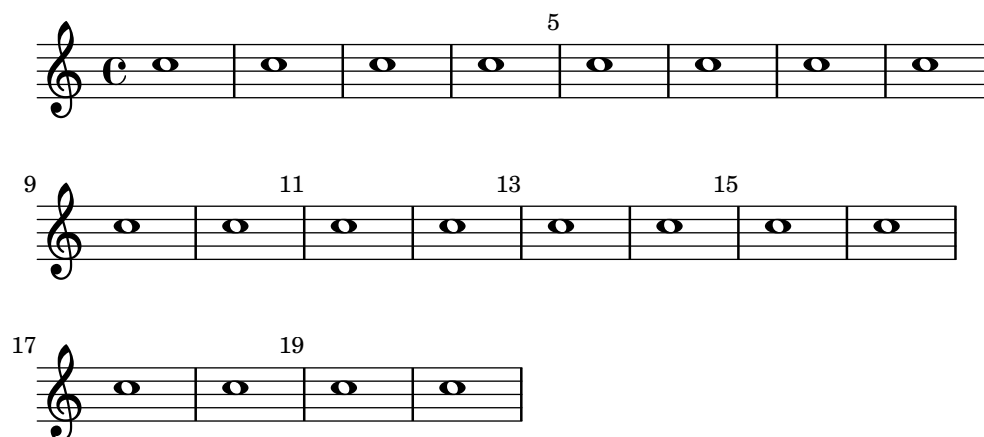
L'intervallo dei numeri di battuta può essere modificato cambiando la funzione di contesto `set-bar-number-visibility`.

```
\relative c' {
```

```

\override Score.BarNumber.break-visibility = #end-of-line-invisible
\context Score \applyContext #(set-bar-number-visibility 4)
\repeat unfold 10 c'1
\context Score \applyContext #(set-bar-number-visibility 2)
\repeat unfold 10 c
}

```



*Numeri di battuta racchiusi in rettangoli o cerchi*

I numeri di battuta possono apparire anche all'interno di rettangoli o cerchi.

```

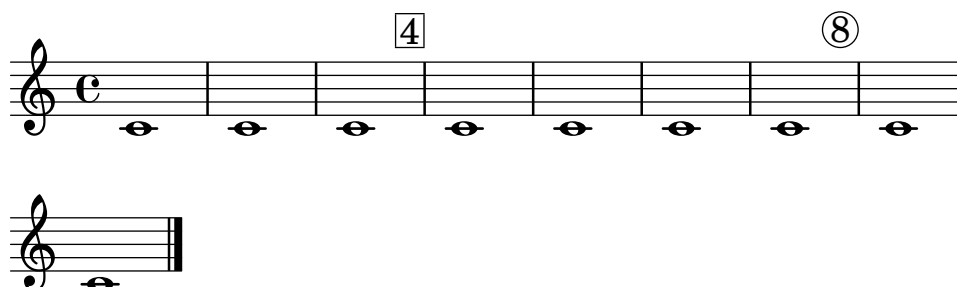
\relative c' {
  % Prevent bar numbers at the end of a line and permit them elsewhere
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 4)

  % Increase the size of the bar number by 2
  \override Score.BarNumber.font-size = #2

  % Draw a box round the following bar number(s)
  \override Score.BarNumber.stencil
    = #(make-stencil-boxer 0.1 0.25 ly:text-interface::print)
  \repeat unfold 5 { c1 }

  % Draw a circle round the following bar number(s)
  \override Score.BarNumber.stencil
    = #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
  \repeat unfold 4 { c1 } \bar "|"
}

```



*Numeri di battuta alternativi*

Si possono impostare due metodi alternativi di numerazione della battuta, utili specialmente per le ripetizioni.

```
\relative c'{
  \set Score.alternativeNumberingStyle = #'numbers
  \repeat volta 3 { c4 d e f | }
  \alternative {
    { c4 d e f | c2 d \break }
    { f4 g a b | f4 g a b | f2 a | \break }
    { c4 d e f | c2 d }
  }
  c1 \break
  \set Score.alternativeNumberingStyle = #'numbers-with-letters
  \repeat volta 3 { c,4 d e f | }
  \alternative {
    { c4 d e f | c2 d \break }
    { f4 g a b | f4 g a b | f2 a | \break }
    { c4 d e f | c2 d }
  }
  c1
}
```

The image displays a musical score with six staves, each representing a different system of notation. The first staff is labeled '1.' and shows a sequence of notes. The second staff is labeled '2' and shows a sequence of notes. The third staff is labeled '3' and shows a sequence of notes. The fourth staff is labeled '5' and shows a sequence of notes. The fifth staff is labeled '6b' and shows a sequence of notes. The sixth staff is labeled '6c' and shows a sequence of notes.

#### *Allineare i numeri di battuta*

Per impostazione predefinita i numeri di battuta sono allineati a destra rispetto al loro oggetto genitore. Di solito si tratta del margine sinistro della linea oppure, se i numeri appaiono

all'interno della linea, del lato sinistro della stanghetta. I numeri possono essere posizionati anche direttamente sopra la stanghetta oppure allineati a sinistra della stanghetta.

```
\relative c' {
  \set Score.currentBarNumber = #111
  \override Score.BarNumber.break-visibility = #all-visible
  % Increase the size of the bar number by 2
  \override Score.BarNumber.font-size = #2
  % Print a bar number every second measure
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
  c1 | c1
  % Center-align bar numbers
  \override Score.BarNumber.self-alignment-X = #CENTER
  c1 | c1
  % Left-align bar numbers
  \override Score.BarNumber.self-alignment-X = #LEFT
  c1 | c1
}
```

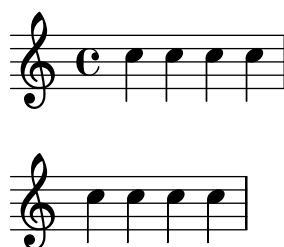


*Togliere i numeri di battuta da uno spartito*

I numeri di battuta possono essere tolti rimuovendo l'incisore `Bar_number_engraver` dal contesto `Score`.

```
\layout {
  \context {
    \Score
    \omit BarNumber
    % or:
    %\remove "Bar_number_engraver"
  }
}

\relative c'' {
  c4 c c c \break
  c4 c c c
}
```



## Vedi anche

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “BarNumber” in *Guida al Funzionamento Interno*, Sezione “Bar\_number\_engraver” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

I numeri di battuta possono collidere con la parte superiore della parentesi quadra di `StaffGroup`, se presente. Per evitare la collisione, si può usare la proprietà `padding` di `BarNumber` per posizionare correttamente il numero. Si veda Sezione “`StaffGroup`” in *Guida al Funzionamento Interno* e Sezione “`BarNumber`” in *Guida al Funzionamento Interno* per maggiori informazioni.

## Controlli di battuta e del numero di battuta

I controlli di battuta aiutano a rilevare gli errori di durata. Il controllo di battuta si inserisce col simbolo della barra verticale, `|`, in un qualsiasi punto in cui è previsto l’inserimento di una stanghetta. Se vengono trovati controlli di battuta in punti diversi, viene creata una lista di avvisi nel file di log che mostra i numeri di linea e le linee in cui il controllo è fallito. Nell’esempio seguente il secondo controllo di battuta segnerà un errore.

```
\time 3/4 c2 e4 | g2 |
```

Una durata non corretta può generare uno spartito completamente alterato, specialmente nel caso di brani polifonici. Quindi il primo passo da compiere per correggere l’input è la verifica dei controlli di battuta e delle durate errate.

Se i controlli di battuta successivi sono spostati dello stesso intervallo musicale, viene mostrato solo il primo messaggio di avviso. Così l’avvertimento si concentra sulla causa dell’errore di tempo.

I controlli di battuta possono essere usati anche all’interno del testo vocale:

```
\lyricmode {
  \time 2/4
  Twin -- kle | Twin -- kle |
}
```

Attenzione: i segni di controllo di ottava nel testo vocale sono elaborati nel momento musicale in cui la sillaba *che segue* il segno di controllo viene elaborata. Se il testo è associato alle note di una voce che ha una pausa all’inizio di una battuta, non è possibile individuare alcuna sillaba all’inizio di quella battuta e apparirà un avvertimento se viene posto un controllo di battuta in quel punto del testo vocale.

È anche possibile ridefinire l’azione da prendere quando si incontra un controllo di battuta o simbolo di barra verticale, `|`, nell’input, in modo che avvenga qualcosa di diverso dal controllo di battuta. Si può fare assegnando un’espressione musicale a `"|"`. Nell’esempio seguente `|`, invece di controllare la fine di una battuta, viene usato per inserire una stanghetta doppia ovunque appaia nell’input.

```
"|" = \bar "||"
{
  c'2 c' |
  c'2 c'
  c'2 | c'
  c'2 c'
}
```



Quando si copiano brani di una certa ampiezza, può essere d’aiuto verificare che i numeri di battuta di LilyPond corrispondano all’originale a partire dal quale si sta scrivendo il brano. Si può abilitare con `\barNumberCheck`, ad esempio,

```
\barNumberCheck #123
```



genererà un avvertimento se `currentBarNumber` non è 123 nel momento in cui viene elaborato.

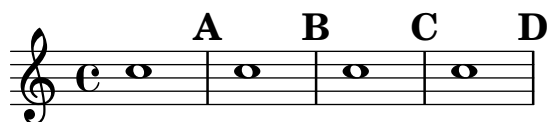
## Vedi anche

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

## Segni di chiamata

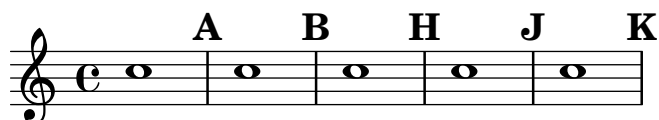
Per creare un segno di chiamata si usa il comando `\mark`.

```
\relative c'' {
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
}
```



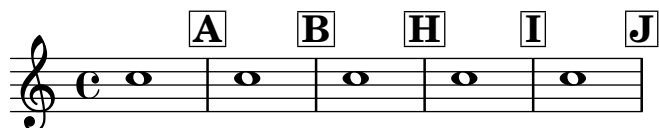
Il segno viene incrementato automaticamente se si usa `\mark \default`, ma è possibile usare anche un numero intero come argomento in modo da impostare il segno manualmente. Il valore da usare viene salvato nella proprietà `rehearsalMark`.

```
\relative c'' {
  c1 \mark \default
  c1 \mark \default
  c1 \mark #8
  c1 \mark \default
  c1 \mark \default
}
```



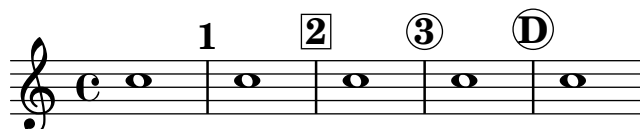
La lettera ‘I’ viene saltata, come vuole la tradizione tipografica. Se si desidera includere la lettera ‘I’, si può usare uno dei seguenti comandi, a seconda dello stile che si vuole (solo lettere, lettere in un quadrato o lettere in un cerchio).

```
\set Score.markFormatter = #format-mark-alphabet
\set Score.markFormatter = #format-mark-box-alphabet
\set Score.markFormatter = #format-mark-circle-alphabet
\relative c'' {
  \set Score.markFormatter = #format-mark-box-alphabet
  c1 \mark \default
  c1 \mark \default
  c1 \mark #8
  c1 \mark \default
  c1 \mark \default
}
```



Lo stile viene definito dalla proprietà `markFormatter`. È una funzione che accoglie come argomenti il segno corrente (un numero intero) e il contesto corrente. Dovrebbe restituire un oggetto testuale. Nell'esempio seguente, `markFormatter` viene prima impostato su una procedura predefinita e dopo alcune misure su una procedura che produce un numero racchiuso in un quadrato.

```
\relative c' {
  \set Score.markFormatter = #format-mark-numbers
  c1 \mark \default
  c1 \mark \default
  \set Score.markFormatter = #format-mark-box-numbers
  c1 \mark \default
  \set Score.markFormatter = #format-mark-circle-numbers
  c1 \mark \default
  \set Score.markFormatter = #format-mark-circle-letters
  c1
}
```



Il file `scm/translation-functions.scm` contiene le definizioni di `format-mark-numbers` (il formato predefinito), `format-mark-box-numbers`, `format-mark-letters` e `format-mark-box-letters`. Possono essere usate come fonte di ispirazione per creare altre funzioni di formattazione.

Si possono usare `format-mark-barnumbers`, `format-mark-box-barnumbers` e `format-mark-circle-barnumbers` per ottenere i numeri di battuta invece di numeri o lettere crescenti.

Si possono specificare manualmente altri stili di segni di chiamata:

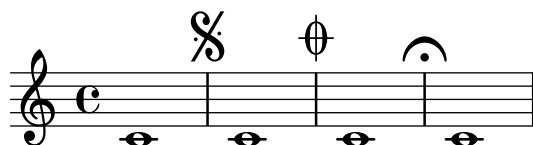
```
\mark "A1"
```

Si noti che `Score.markFormatter` non ha effetto sui segni specificati in questo modo. Tuttavia, è possibile applicare un `\markup` alla stringa.

```
\mark \markup{ \box A1 }
```

I glifi musicali (come il Segno) possono essere posti dentro il comando `\mark`

```
\relative c' {
  c1 \mark \markup { \musicglyph #"scripts.segno" }
  c1 \mark \markup { \musicglyph #"scripts.coda" }
  c1 \mark \markup { \musicglyph #"scripts.ufermata" }
  c1
}
```



L'elenco dei simboli che possono essere prodotti con `\musicglyph` si trova in `<undefined>` [The Feta font], pagina `<undefined>`.

Per le più comuni modifiche relative al posizionamento dei segni di chiamata, si veda `<undefined>` [Formatting text], pagina `<undefined>`. Per ottenere un controllo più preciso si consiglia di

studiare il funzionamento della proprietà `break-alignable-interface` descritta in Sezione 5.5.1 [Aligning objects], pagina 627.

Il file `scm/translation-functions.scm` contiene le definizioni di `format-mark-numbers` e `format-mark-letters`, che possono essere usate come fonte di ispirazione per creare altre funzioni di formattazione.

## Vedi anche

Guida alla notazione: `<undefined>` [The Feta font], pagina `<undefined>`, `<undefined>` [Formatting text], pagina `<undefined>`, Sezione 5.5.1 [Aligning objects], pagina 627.

File installati: `scm/translation-functions.scm`.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “MarkEvent” in *Guida al Funzionamento Interno*, Sezione “Mark\_engraver” in *Guida al Funzionamento Interno*, Sezione “RehearsalMark” in *Guida al Funzionamento Interno*.

## 1.2.6 Questioni ritmiche particolari

### Abbellimenti

Gli abbellimenti sono degli ornamenti musicali che hanno un carattere in corpo più piccolo e non alterano la durata della misura.

```
\relative {
  c''4 \grace b16 a4(
  \grace { b16 c16 } a2)
}
```



Esistono altri tre tipi di abbellimenti possibili; l'*acciaccatura* – un abbellimento in tempo libero indicato da una nota con legatura di portamento e un gambo barrato – e l'*appoggiatura*, che sottrae un valore determinato della nota principale a cui corrisponde e ha un gambo non barrato. È anche possibile creare un abbellimento con gambo barrato, come l'*acciaccatura*, ma privo di legatura di portamento, in modo da collocarla tra note già poste sotto una legatura: si usa il comando `\slashedGrace`.

```
\relative {
  \acciaccatura d''8 c4
  \appoggiatura e8 d4
  \acciaccatura { g16 f } e2
  \slashedGrace a,8 g4
  \slashedGrace b16 a4(
  \slashedGrace b8 a2)
}
```



Il posizionamento degli abbellimenti è sincronizzato sui diversi righi. Nell'esempio seguente, ci sono due abbellimenti da un sedicesimo ogni abbellimento da un ottavo

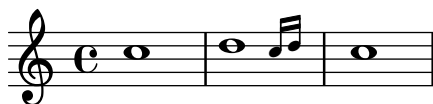
<<

```
\new Staff \relative { e''2 \grace { c16 d e f } e2 }
\new Staff \relative { c''2 \grace { g8 b } c2 }
>>
```



Se si desidera risolvere una nota su un abbellimento, si usa il comando `\afterGrace`. Considera due argomenti: la nota principale e gli abbellimenti che la seguono.

```
\relative { c''1 \afterGrace d1 { c16[ d] } c1 }
```



In questo modo, gli abbellimenti sono collocati dopo uno spazio corrispondente a  $3/4$  della durata della nota principale. La frazione predefinita  $3/4$  può essere modificata attraverso `afterGraceFraction`. L'esempio seguente mostra le diverse spazature che si ottengono con la frazione predefinita, con  $15/16$  e infine con  $1/2$  della nota principale.

```
<<
\new Staff \relative {
  c''1 \afterGrace d1 { c16[ d] } c1
}
\new Staff \relative {
  #(define afterGraceFraction (cons 15 16))
  c''1 \afterGrace d1 { c16[ d] } c1
}
\new Staff \relative {
  #(define afterGraceFraction (cons 1 2))
  c''1 \afterGrace d1 { c16[ d] } c1
}
>>
```



Lo spazio tra la nota principale e l'abbellimento può essere definito anche attraverso delle pause spaziatrici. L'esempio seguente sposta l'abbellimento di uno spazio equivalente ai  $7/8$  della nota principale.

```
\new Voice \relative {
```

```

<<
  { d''1^\trill_( }
  { s2 s4. \grace { c16 d } }
>>
c1)
}

```



L'espressione musicale introdotta dal comando `\grace` avrà delle impostazioni tipografiche speciali; per esempio, per rimpicciolire il tipo di carattere e impostare le direzioni. Dunque le modifiche che sovrascrivono tali impostazioni speciali devono essere poste all'interno del blocco `\grace`. Lo stesso vale per le modifiche che ripristinano i valori predefiniti. Nell'esempio seguente la direzione predefinita del gambo viene prima sovrascritta e poi ripristinata.

```

\new Voice \relative {
  \acciaccatura {
    \stemDown
    f''16->
    \stemNeutral
  }
  g4 e c2
}

```



## Frammenti di codice selezionati

*Usare il gambo barrato degli abbellimenti con le teste normali*

Il gambo barrato presente nelle acciaccature può essere applicato in altre situazioni.

```

\relative c'' {
  \override Flag.stroke-style = #"grace"
  c8( d2) e8( f4)
}

```



*Modificare l'aspetto degli abbellimenti di un intero brano*

L'aspetto di tutte le espressioni contenute nei blocchi `\grace` di un brano può essere modificato con le funzioni `add-grace-property` e `remove-grace-property`. L'esempio seguente toglie la definizione della direzione di `Stem` nell'abbellimento, in modo che gli abbellimenti non siano sempre rivolti in su, e barra le teste di nota.

```

\relative c'' {
  \new Staff {
    $(remove-grace-property 'Voice 'Stem 'direction)
    $(add-grace-property 'Voice 'NoteHead 'style 'cross)
  }
}

```

```

\new Voice {
  \acciaccatura { f16 } g4
  \grace { d16 e } f4
  \appoggiatura { f,32 g a } e2
}
}
}

```



### *Ridefinire le impostazioni predefinite globali degli abbellimenti*

Le impostazioni globali predefinite degli abbellimenti sono salvate negli identificatori `startGraceMusic`, `stopGraceMusic`, `startAcciaccaturaMusic`, `stopAcciaccaturaMusic`, `startAppoggiaturaMusic` e `stopAppoggiaturaMusic`, che sono definiti nel file `ly/grace-init.ly`. Ridefinendoli si possono ottenere effetti diversi.

```

startAcciaccaturaMusic = {
  <>(
    \override Flag.stroke-style = #"grace"
    \slurDashed
  )
}

stopAcciaccaturaMusic = {
  \revert Flag.stroke-style
  \slurSolid
  <>)
}

\relative c' {
  \acciaccatura d8 c1
}

```



### *Posizionare gli abbellimenti con dello spazio fluttuante*

Se si imposta la proprietà '`strict-grace-spacing`', le colonne musicali degli abbellimenti 'fluttuano', ovvero si scollegano dalle note normali: prima vengono spaziate le note normali, poi le colonne musicali degli abbellimenti vengono messe a sinistra delle colonne delle note principali.

```

\relative c' {
  <<
    \override Score.SpacingSpanner.strict-grace-spacing = ##t
    \new Staff \new Voice {
      \afterGrace c4 { c16[ c8 c16] }
      c8[ \grace { b16 d } c8]
      c4 r
    }
    \new Staff {
      c16 c c c c c c c c4 r
    }
  >>
}

```



## Vedi anche

Glossario musicale: Sezione “acciaccatura” in *Glossario Musicale*, Sezione “acciaccatura” in *Glossario Musicale*, Sezione “appoggiatura” in *Glossario Musicale*.

Guida alla notazione: [Scaling durations](#), pagina [Manual beams](#), pagina [Manual beams](#).

File installati: `ly/grace-init.ly`.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “GraceMusic” in *Guida al Funzionamento Interno*, Sezione “Grace\_beam\_engraver” in *Guida al Funzionamento Interno*, Sezione “Grace\_auto\_beam\_engraver” in *Guida al Funzionamento Interno*, Sezione “Grace\_engraver” in *Guida al Funzionamento Interno*, Sezione “Grace\_spacing\_engraver” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Una *acciaccatura* con molte note raggruppate sotto una travatura è priva della barra trasversale e ha il medesimo aspetto di una *appoggiatura* composta da varie note raggruppate sotto una travatura.

La sincronizzazione degli abbellimenti può nascondere delle sorprese, perché vengono sincronizzati anche altri elementi della notazione del rigo, come le armature di chiave, le stanghette, etc. Fai attenzione quando metti insieme righe che hanno degli abbellimenti con righe che non ne hanno. Ad esempio

```
<<
\new Staff \relative { e''4 \bar ".|:" \grace c16 d2. }
\new Staff \relative { c''4 \bar ".|:" d2. }
>>
```



Si può ovviare a questo problema inserendo degli abbellimenti della durata corrispondente negli altri righe. Riprendendo l'esempio precedente

```
<<
\new Staff \relative { e''4 \bar ".|:" \grace c16 d2. }
\new Staff \relative { c''4 \bar ".|:" \grace s16 d2. }
>>
```

&gt;&gt;



L'uso del comando `\grace` nella parte con le pause spaziatrici è obbligatorio, anche se la parte visibile usa `\acciaccatura` o `\appoggiatura`, perché altrimenti apparirà un orribile frammento di legatura di portamento che connette la nota di abbellimento invisibile alla nota seguente.

Le sezioni con abbellimenti devono essere usate solamente all'interno di espressioni musicali sequenziali. Non è permesso annidare o affiancare gruppi di abbellimenti; potrebbero verificarsi blocchi del programma o altri errori se non si rispetta questa limitazione.

Ogni abbellimento generato nell'output MIDI ha una durata di  $1/4$  della sua vera durata. Se la durata complessiva degli abbellimenti è maggiore della durata della nota che li precede, verrà generato l'errore "Going back in MIDI time". A meno che non si diminuisca la durata degli abbellimenti, ad esempio:

```
c'8 \acciaccatura { c'8[ d' e' f' g'] }
```

diventa:

```
c'8 \acciaccatura { c'16[ d' e' f' g'] }
```

Oppure si cambia esplicitamente la durata musicale:

```
c'8 \acciaccatura { \scaleDurations 1/2 { c'8[ d' e' f' g'] } }
```

Vedi `\undefined` [Scaling durations], pagina `\undefined`.

## Allineamento sulle cadenze

Nell'ambito di una partitura per orchestra, le cadenze presentano un problema peculiare: quando si scrive una partitura che include una cadenza o un altro passaggio solistico, tutti gli altri strumenti devono saltare esattamente la durata complessiva delle note del passaggio, altrimenti inizieranno troppo presto o troppo tardi.

Una possibile soluzione a questo problema consiste nell'uso delle funzioni `mmrest-of-length` e `skip-of-length`. Queste funzioni Scheme prendono come argomento una sezione di musica salvata in una variabile e generano una pausa multipla o `\skip` della lunghezza esatta del brano.

```
MyCadenza = \relative {
  c'4 d8 e f g g4
  f2 g4 g
}

\new GrandStaff <<
  \new Staff {
    \MyCadenza c'1
    \MyCadenza c'1
  }
  \new Staff {
    #(mmrest-of-length MyCadenza)
    c'1
    #(skip-of-length MyCadenza)
    c'1
  }
>>
```





## Vedi anche

Glossario musicale: Sezione “cadenza” in *Glossario Musicale*.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

## Gestione del tempo

Il tempo è gestito da `Timing_translator`, che si trova nel contesto `Score`. Un suo alias, `Timing`, viene aggiunto nel contesto nel quale si trova `Timing_translator`. Per assicurarsi che l’alias `Timing` sia disponibile, occorre istanziare esplicitamente il contesto che lo contiene (come `Voice` o `Staff`).

Si usano le seguenti proprietà di `Timing` per tenere traccia del tempo in una partitura.

### `currentBarNumber`

Il numero di battuta corrente. Un esempio che mostra l’uso di questa proprietà si trova in `<undefined> [Bar numbers]`, pagina `<undefined>`.

### `measureLength`

La durata delle misure nel tempo corrente. Per un tempo di 4/4 è 1, per un tempo di 6/8 è 3/4. Il suo valore determina quando debbano essere inserite le stanghette e come debbano essere generate le travature automatiche.

### `measurePosition`

Il punto della misura in cui ci si trova. Questa quantità viene reimpostata sottraendo `measureLength` ogni volta che `measureLength` viene raggiunto o superato. Quando questo accade, `currentBarNumber` viene incrementato.

`timing` Se impostato su `#t`, le variabili precedenti sono aggiornate ad ogni momento temporale. Se impostato su `#f`, l’incisore rimane nella misura corrente per un tempo indefinito.

Si può cambiare il tempo impostando esplicitamente una qualsiasi di queste variabili. Nel prossimo esempio, viene visualizzata l’indicazione di tempo predefinita di 4/4, ma `measureLength` è impostato su 5/4. A 4/8 della terza misura, `measurePosition` si sposta in avanti di 1/8 fino a 5/8, diminuendo quella misura di 1/8. Quindi la stanghetta successiva si troverà a 9/8 invece che a 5/4.

```
\new Voice \relative {
  \set Timing.measureLength = #(ly:make-moment 5/4)
  c'1 c4 |
  c1 c4 |
  c4 c
  \set Timing.measurePosition = #(ly:make-moment 5/8)
  b4 b b8 |
  c4 c1 |
}
```



Come mostra l'esempio, `ly:make-moment n m` definisce una durata di  $n/m$  della nota intera. Ad esempio, `ly:make-moment 1 8` corrisponde alla durata di un ottavo mentre `ly:make-moment 7 16` a quella di sette sedicesimi.

## Vedi anche

Guida alla notazione: `<undefined>` [Bar numbers], pagina `<undefined>`, `<undefined>` [Unmetered music], pagina `<undefined>`.

Frammenti di codice: Sezione “Rhythms” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Timing translator” in *Guida al Funzionamento Interno*, Sezione “Score” in *Guida al Funzionamento Interno*.

## 1.3 Segni di espressione

RONDO  
*Allegro*

Questa sezione elenca vari segni di espressione che si possono usare in una partitura.

### 1.3.1 Segni di espressione collegati alle note

Questa sezione spiega come creare segni di espressione collegati alle note: articolazioni, abbellimenti e dinamiche. Sono trattati anche i metodi per creare nuove indicazioni dinamiche.

#### Articolazioni e abbellimenti

I diversi simboli che rappresentano articolazioni, ornamenti e altre indicazioni esecutive possono essere collegati a una nota con questa sintassi:

`nota\nome`

I valori possibili per *nome* sono elencati in [\[List of articulations\]](#), pagina [\[undefined\]](#). Ad esempio:

```
\relative {
  c''4\staccato c\mordent b2\turn
  c1\fermata
}
```



Alcune di queste articolazioni hanno delle abbreviazioni che ne semplificano l’inserimento. Le abbreviazioni sono attaccate al nome della nota e la loro sintassi è composta da un trattino - seguito da un simbolo che indica l’articolazione. Esistono abbreviazioni predefinite per *marcato*, *chiuso*, *tenuto*, *staccatissimo*, *accento*, *staccato* e *portato*. L’output corrispondente è:

```
\relative {
  c''4-^ c-+ c-- c-!
  c4-> c-. c2-_
}
```



Le regole per il posizionamento predefinito delle articolazioni sono definite in `scm/script.scm`. Articolazioni e ornamenti possono essere posizionati manualmente sopra o sotto il rigo; si veda Sezione 5.4.2 [\[Direction and placement\]](#), pagina 611.

Le articolazioni sono oggetti `Script`. Le loro proprietà sono descritte in dettaglio in Sezione “Script” in *Guida al Funzionamento Interno*.

Le articolazioni possono essere attaccate alle pause e alle note ma non alle pause multiple. Esiste un comando speciale predefinito, `\fermataMarkup`, che permette di attaccare un segno di corona a una pausa multipla (e soltanto ad essa). Questo crea un oggetto `MultiMeasureRestText`.

```
\override Script.color = #red
\override MultiMeasureRestText.color = #blue
a'2\fermata r\fermata
R1\fermataMarkup
```



Oltre alle articolazioni, si può attaccare alle note anche un testo, posto tra virgolette o in un blocco `\markup{}`. Si veda [\[Text scripts\]](#), pagina [\[undefined\]](#).

Ulteriori informazioni sull’ordine degli oggetti `Script` e `TextScript` collegati alle note si trovano in Sezione “Posizionamento degli oggetti” in *Manuale di Apprendimento*.

## Frammenti di codice selezionati

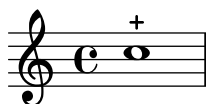
*Modificare i valori predefiniti per le abbreviazioni delle articolazioni*

Le abbreviazioni sono definite in ‘ly/script-init.ly’, dove sono assegnati valori predefiniti alle variabili `dashHat`, `dashPlus`, `dashDash`, `dashBang`, `dashLarger`, `dashDot` e `dashUnderscore`. Questi valori predefiniti possono essere modificati. Ad esempio, per associare l’abbreviazione `-+` (`dashPlus`) al simbolo del trillo invece che al simbolo `+` predefinito, si assegna il valore `trill` alla variabile `dashPlus`:

```
\relative c'' { c1-+ }
```

```
dashPlus = "trill"
```

```
\relative c'' { c1-+ }
```



*Controllo dell’ordine verticale degli script*

L’ordine verticale degli script è determinato dalla proprietà `'script-priority`. Più il numero è piccolo, più sarà posto vicino alla nota. In questo esempio, il simbolo di diesis (oggetto `TextScript`) ha prima la priorità più bassa, dunque è posto più in basso nel primo esempio. Nel secondo, il trillo (oggetto `Script`) ha la priorità più bassa, quindi si trova all’interno. Quando due oggetti hanno la stessa priorità, l’ordine in cui sono inseriti determina quale viene prima.

```
\relative c''' {
  \once \override TextScript.script-priority = #-100
  a2^\prall^\markup { \sharp }

  \once \override Script.script-priority = #-100
  a2^\prall^\markup { \sharp }
}
```



*Creare un gruppetto ritardato*

Creare un gruppetto ritardato, dove la nota più bassa del gruppetto usa l’alterazione, richiede vari `\override`. La proprietà `outside-staff-priority` deve essere impostata su `#f`, perché altrimenti questa avrebbe la precedenza sulla proprietà `avoid-slur`. Cambiando le frazioni `2/3` e `1/3` si aggiusta la posizione orizzontale.

```
\relative c'' {
  c2*2/3 ( s2*1/3\turn d4) r
  <<
  { c4.( d8) }
```

```

    { s4 s\turn }
  >>
  \transpose c d \relative c'' <<
    { c4.( d8) }
    {
      s4
      \once \set suggestAccidentals = ##t
      \once \override AccidentalSuggestion.outside-staff-priority = ##f
      \once \override AccidentalSuggestion.avoid-slur = #'inside
      \once \override AccidentalSuggestion.font-size = -3
      \once \override AccidentalSuggestion.script-priority = -1
      \single \hideNotes
      b8-\turn \noBeam
      s8
    }
  >>
}

```



## Vedi anche

Glossario Musicale: Sezione “tenuto” in *Glossario Musicale*, Sezione “accento” in *Glossario Musicale*, Sezione “staccato” in *Glossario Musicale*, Sezione “portato” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Posizionamento degli oggetti” in *Manuale di Apprendimento*.

Guida alla notazione: *<undefined>* [Text scripts], pagina *<undefined>*, Sezione 5.4.2 [Direction and placement], pagina 611, *<undefined>* [List of articulations], pagina *<undefined>*, *<undefined>* [Trills], pagina *<undefined>*.

File installati: `scm/script.scm`.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Script” in *Guida al Funzionamento Interno*, Sezione “TextScript” in *Guida al Funzionamento Interno*.

## Dinamiche

Le indicazioni dinamiche assolute si indicano con un comando che segue una nota, come ad esempio `c4\ff`. Le indicazioni dinamiche disponibili sono `\ppppp`, `\pppp`, `\ppp`, `\pp`, `\p`, `\mp`, `\mf`, `\f`, `\ff`, `\fff`, `\ffff`, `\ffffff`, `\fp`, `\sf`, `\sff`, `\sp`, `\spp`, `\sfz` e `\rfz`. Le indicazioni dinamiche possono essere posizionate manualmente sopra o sotto il rigo, come è spiegato in dettaglio in Sezione 5.4.2 [Direction and placement], pagina 611.

```

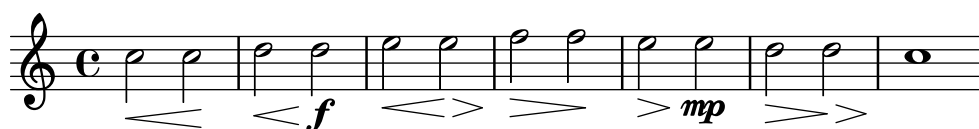
\relative c'' {
  c2\ppp c\mp
  c2\rfz c^\mf
  c2_\spp c^\ff
}

```



Un'indicazione di *crescendo* inizia con `\<` e termina con `\!`, un'indicazione dinamica assoluta o un'ulteriore indicazione di *crescendo* o *decrescendo*. Un'indicazione di *decrescendo* inizia con `\>` e termina nello stesso modo, ovvero con `\!`, un'indicazione dinamica assoluta oppure un altro segno di *crescendo* o *decrescendo*. Si possono usare `\cr` e `\decr` al posto di `\<` e `\>`. Le *forcelle* vengono create con questa notazione.

```
\relative c'' {
  c2\< c\!
  d2\< d\!f
  e2\< e\>
  f2\> f\!
  e2\> e\mp
  d2\> d\>
  c1\!
}
```



Una forcella che termina con `\!` si estenderà fino al margine destro della nota a cui è assegnato `\!`. Nel caso in cui sia terminata con l'inizio di un altro segno di *crescendo* o *decrescendo*, si estenderà fino al centro della nota a cui è assegnato il successivo `\<` o `\>`. La forcella successiva partirà dal margine destro della stessa nota invece che dal margine sinistro, come accade quando si termina con `\!`.

```
\relative {
  c''1\< | c4 a c\< a | c4 a c\! a\< | c4 a c a\!
}
```



Le forcelle terminate con indicazioni dinamiche assolute invece che da `\!` avranno un aspetto simile. Tuttavia, la lunghezza dell'indicazione dinamica assoluta stessa può cambiare il punto in cui finisce la forcella precedente.

```
\relative {
  c''1\< | c4 a c\mf a | c1\< | c4 a c\ffff a
}
```



Occorre usare le pause spaziatrici per attaccare più di un'indicazione a una nota. Questo è utile soprattutto quando si aggiunge un *crescendo* e un *decrescendo* alla stessa nota:

```
\relative {
  c''4\< c\! d\> e\!
```

```
<< f1 { s4 s4\< s4\> s4\! } >>
}
```



Il comando `\espressivo` permette di indicare un crescendo e un decrescendo sulla stessa nota. Tuttavia, si tenga presente che viene implementato come articolazione, non come dinamica.

```
\relative {
  c'2 b4 a
  g1\espressivo
}
```



Le indicazioni di crescendo testuali iniziano con `\cresc`, quelle di decrescendo con `\decre` o `\dim`. Le linee di estensione sono aggiunte automaticamente.

```
\relative {
  g'8\cresc a b c b c d e\mf |
  f8\decre e d c e\> d c b |
  a1\dim ~ |
  a2. r4\! |
}
```



Le indicazioni testuali per i cambi di dinamica possono essere impiegate anche per sostituire le forcelle:

```
\relative c' {
  \crescTextCresc
  c4\< d e f\! |
  \dimTextDecresc
  g4\> e d c\! |
  \dimTextDecr
  e4\> d c b\! |
  \dimTextDim
  d4\> c b a\! |
  \crescHairpin
  \dimHairpin
  c4\< d\! e\> d\! |
}
```



Per creare nuove indicazioni dinamiche assolute o testi da allineare alle dinamiche, si veda [\[New dynamic marks\]](#), pagina [\[undefined\]](#).

Il posizionamento verticale della dinamica è gestito da Sezione “DynamicLineSpanner” in *Guida al Funzionamento Interno*.

Esiste un contesto `Dynamics` che permette di posizionare le indicazioni dinamiche su un'apposita linea orizzontale. Si usano le pause spaziatrici per indicarne la collocazione temporale (le note in un contesto `Dynamics` occupano infatti il rispettivo valore musicale, ma senza comparire sul rigo). Il contesto `Dynamics` può contenere altri elementi utili come indicazioni testuali, estensori del testo e indicazioni di pedalizzazione del pianoforte.

```
<<
\new Staff \relative {
  c'2 d4 e |
  c4 e e,2 |
  g'4 a g a |
  c1 |
}
\new Dynamics {
  s1\< |
  s1\f |
  s2\dim s2-"rit." |
  s1\p |
}
>>
```



## Comandi predefiniti

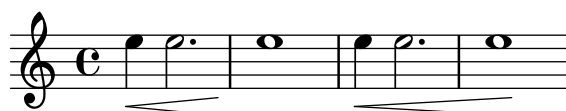
`\dynamicUp`, `\dynamicDown`, `\dynamicNeutral`, `\crescTextCresc`, `\dimTextDim`, `\dimTextDecr`, `\dimTextDecresc`, `\crescHairpin`, `\dimHairpin`.

## Frammenti di codice selezionati

*Impostare il comportamento delle forcelle sulle stanghette*

Se la nota che termina una forcina si trova sul primo battito di una battuta, la forcina si ferma prima della stanghetta che precede la nota. Si può controllare questo comportamento modificando la proprietà `'to-barline`.

```
\relative c'' {
  e4\< e2.
  e1\!
  \override Hairpin.to-barline = ##f
  e4\< e2.
  e1\!
}
```

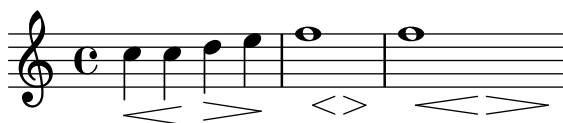




*Impostare la lunghezza minima delle forcelle*

Se le forcelle sono troppo corte, possono essere allungate modificando la proprietà `minimum-length` dell'oggetto `Hairpin`.

```
\relative c'' {
  c4\< c\! d\> e\!
  << f1 { s4 s\< s\> s\! } >>
  \override Hairpin.minimum-length = #5
  << f1 { s4 s\< s\> s\! } >>
}
```

*Forcelle con notazione al niente*

Le forcelle di dinamica possono essere rappresentate con una punta tonda (notazione “al niente”) impostando la proprietà `circled-tip` dell'oggetto `Hairpin` su `#t`.

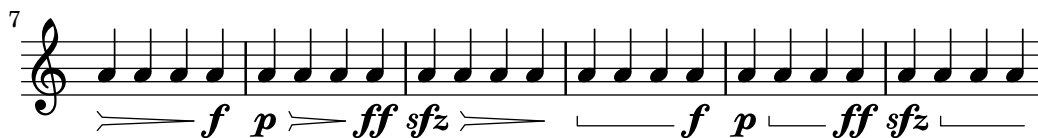
```
\relative c'' {
  \override Hairpin.circled-tip = #t
  c2\< c\!
  c4\> c\< c2\!
}
```

*Stampare le forcelle in vari stili*

Il segno di dinamica della forcilla può avere diversi stili

```
\relative c'' {
  \override Hairpin.stencil = #flared-hairpin
  a4\< a a a\f
  a4\p\< a a a\ff
  a4\sfz\< a a a\!
  \override Hairpin.stencil = #constante-hairpin
  a4\< a a a\f
  a4\p\< a a a\ff
  a4\sfz\< a a a\!
  \override Hairpin.stencil = #flared-hairpin
  a4\> a a a\f
  a4\p\> a a a\ff
  a4\sfz\> a a a\!
  \override Hairpin.stencil = #constante-hairpin
  a4\> a a a\f
  a4\p\> a a a\ff
  a4\sfz\> a a a\!
}
```





### *Dinamiche e segni testuali allineati verticalmente*

Tutti gli oggetti `DynamicLineSpanner` (forcelle e testi di dinamica) sono posti a una distanza minima dal rigo determinata da `'staff-padding`. Se si imposta `'staff-padding` su un valore abbastanza grande, le dinamiche saranno allineate.

```
music = \relative c' {
  a'2\p b\f
  e4\p f\f> g, b\p
  c2^\markup { \huge gorgeous } c^\markup { \huge fantastic }
}

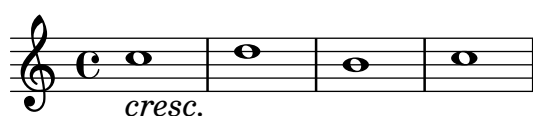
{
  \music
  \break
  \override DynamicLineSpanner.staff-padding = #3
  \textLengthOn
  \override TextScript.staff-padding = #1
  \music
}
```



### *Nascondere la linea di estensione per le dinamiche testuali*

I cambi di dinamica in stile testuale (come *cresc.* e *dim.*) appaiono con una linea tratteggiata che mostra la loro estensione. Questa linea può essere soppressa nel modo seguente:

```
\relative c'' {
  \override DynamicTextSpanner.style = #'none
  \crescTextCresc
  c1\< | d | b | c\!
}
```



### *Nascondere la linea di estensione per le dinamiche testuali*

Il testo usato per i crescendo e i decrescendo può essere cambiato modificando le proprietà di contesto `crescendoText` e `decrescendoText`.

Lo stile della linea dell'estensore può essere cambiato modificando la proprietà `'style` di `DynamicTextSpanner`. Il valore predefinito è `'dashed-line`; gli altri valori possibili sono `'line`, `'dotted-line` e `'none`.

```
\relative c'' {
  \set crescendoText = \markup { \italic { cresc. poco } }
  \set crescendoSpanner = #'text
  \override DynamicTextSpanner.style = #'dotted-line
  a2\< a
  a2 a
  a2 a
  a2 a\mf
}
```



## Vedi anche

Glossario Musicale: Sezione “al niente” in *Glossario Musicale*, Sezione “crescendo” in *Glossario Musicale*, Sezione “decrescendo” in *Glossario Musicale*, Sezione “forcella” in *Glossario Musicale*. Manuale di apprendimento: Sezione “Articolazione e dinamiche” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.4.2 [Direction and placement], pagina 611, [\[New dynamic marks\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [Enhancing MIDI output], pagina [\[undefined\]](#), [\[undefined\]](#) [Controlling MIDI dynamics], pagina [\[undefined\]](#).

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “DynamicText” in *Guida al Funzionamento Interno*, Sezione “Hairpin” in *Guida al Funzionamento Interno*, Sezione “DynamicLineSpanner” in *Guida al Funzionamento Interno*, Sezione “Dynamics” in *Guida al Funzionamento Interno*.

## Nuove indicazioni dinamiche

Il modo più semplice per creare indicazioni dinamiche è usare gli oggetti `\markup`.

```
moltoF = \markup { molto\dynamic f }

\relative {
  <d' e>16_\moltoF <d e>
  <d e>2..
}
```



In modalità markup, si possono creare dinamiche editoriali (racchiuse tra parentesi normali o quadrate). La sintassi della modalità markup è descritta in [\[undefined\]](#) [Formatting text], pagina [\[undefined\]](#).

```
roundF = \markup {
  \center-align \concat { \bold { \italic ( } }
```

```

\dynamic f \bold { \italic ) } } }
boxF = \markup { \bracket { \dynamic f } }
\relative {
  c'1_\roundF
  c1_\boxF
}

```



È possibile creare semplicemente indicazioni dinamiche centrate verticalmente con la funzione `make-dynamic-script`.

```

sfzp = #(make-dynamic-script "sfzp")
\relative {
  c'4 c c\sfpz c
}

```



In generale, `make-dynamic-script` assume come argomento qualsiasi oggetto markup. Il tipo di carattere per la dinamica contiene solo i caratteri `f`, `m`, `p`, `r`, `s` e `z`; dunque, se si desidera creare un'indicazione dinamica che contenga testo semplice e simboli di punteggiatura, occorre usare dei comandi markup che ripristinino la famiglia e la codifica del tipo di carattere per il testo normale, ad esempio `\normal-text`. Il vantaggio nell'uso di `make-dynamic-script` al posto di un normale markup è l'allineamento verticale degli oggetti markup e delle forcelle collegate alla stessa testa di nota.

```

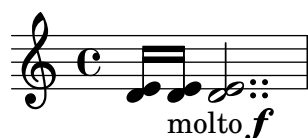
roundF = \markup { \center-align \concat {
  \normal-text { \bold { \italic ( ) } }
  \dynamic f
  \normal-text { \bold { \italic ) } } } }
boxF = \markup { \bracket { \dynamic f } }
mfEspress = \markup { \center-align \line {
  \hspace #3.7 mf \normal-text \italic espress. } }
roundFdynamic = #(make-dynamic-script roundF)
boxFdynamic = #(make-dynamic-script boxF)
mfEspressDynamic = #(make-dynamic-script mfEspress)
\relative {
  c'4_\roundFdynamic\< d e f
  g,1~_\boxFdynamic\>
  g1
  g'1~\mfEspressDynamic
  g1
}

```



Si può usare anche la forma Scheme della modalità markup. La sintassi è spiegata in Sezione “Markup construction in Scheme” in *Estendere*.

```
moltoF = #(make-dynamic-script
           (markup #:normal-text "molto"
                   #:dynamic "f"))
\relative {
  <d' e>16 <d e>
  <d e>2..\moltoF
}
```



Per allineare a sinistra il testo di dinamica invece di centrarlo su una nota, si usa un `\tweak`:

```
moltoF = \tweak DynamicText.self-alignment-X #LEFT
         #(make-dynamic-script
           (markup #:normal-text "molto"
                   #:dynamic "f"))
\relative {
  <d' e>16 <d e>
  <d e>2..\moltoF <d e>1
}
```



Le impostazioni dei tipi di carattere in modalità markup sono descritti in [\[Selecting font and font size\]](#), pagina [\[undefined\]](#).

## Vedi anche

Guida alla notazione: [\[undefined\]](#) [\[Formatting text\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Selecting font and font size\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Enhancing MIDI output\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Controlling MIDI dynamics\]](#), pagina [\[undefined\]](#).

Extending LilyPond: Sezione “Markup construction in Scheme” in *Estendere*.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

### 1.3.2 Indicazioni espressive curvilinee

Questa sezione spiega come creare varie indicazioni espressive con forma curvilinea: legature di portamento, legature di frase, respiri, portamenti indeterminati discendenti (cadute) o ascendenti.

## Legature di portamento

Le *legature di portamento* si inseriscono con delle parentesi:

**Nota:** Nella musica polifonica, una legatura di portamento deve terminare nella stessa voce in cui è iniziata.

```
\relative {
  f''4( g a) a8 b(
  a4 g2 f4)
  <c e>2( <b d>2)
}
```



Le legature di portamento possono essere posizionate manualmente sopra o sotto il rigo, come è spiegato in Sezione 5.4.2 [Direction and placement], pagina 611.

Tracciare due legature di portamento simultanee o sovrapposte richiede una particolare attenzione. Di solito le legature di portamento più esterne indicano in realtà una legatura di frase e le legature di frase possono essere sovrapposte a una normale legatura, vedi [\[Phrasing slurs\]](#), pagina [\[undefined\]](#). Quando invece si vogliono usare molteplici legature di portamento normali in una sola voce, l'inizio e la fine di ogni legatura devono essere preceduti da un `\=` seguito da un numero identificativo o del testo.

```
\fixed c' {
  <c~ f\=1( g\=2( >2 <c e\=1) a\=2) >
}
```



Le legature di portamento possono essere continue, punteggiate o tratteggiate. Lo stile predefinito è quello continuo:

```
\relative {
  c'4( e g2)
  \slurDashed
  g4( e c2)
  \slurDotted
  c4( e g2)
  \slurSolid
  g4( e c2)
}
```



Le legature di portamento possono essere anche semitratteggiate (half-dashed), ovvero con la prima metà tratteggiata e la seconda continua; oppure semicontinue (half-solid), ovvero con la prima metà continua e la seconda tratteggiata:

```
\relative {
```

```

c'4( e g2)
\slurHalfDashed
g4( e c2)
\slurHalfSolid
c4( e g2)
\slurSolid
g4( e c2)
}

```



Si possono definire modelli di tratteggio personalizzati per le legature di portamento:

```

\relative {
  c'4( e g2)
  \slurDashPattern #0.7 #0.75
  g4( e c2)
  \slurDashPattern #0.5 #2.0
  c4( e g2)
  \slurSolid
  g4( e c2)
}

```



## Comandi predefiniti

`\slurUp`, `\slurDown`, `\slurNeutral`, `\slurDashed`, `\slurDotted`, `\slurHalfDashed`, `\slurHalfSolid`, `\slurDashPattern`, `\slurSolid`.

## Frammenti di codice selezionati

*Uso delle doppie legature di portamento per gli accordi legati*

Alcuni compositori scrivono due legature di portamento per indicare gli accordi legati. Si può ottenere questo risultato impostando `doubleSlurs`.

```

\relative c' {
  \set doubleSlurs = ##t
  <c e>4( <d f> <c e> <d f>)
}

```



*Posizionare il testo a margine dentro le legature di portamento*

I testi a margine devono avere la proprietà `outside-staff-priority` impostata su `false` per poter apparire dentro le legature di portamento.

```

\relative c'' {

```

```

\override TextScript.avoid-slur = #'inside
\override TextScript.outside-staff-priority = ##f
c2(^\markup { \halign #-10 \natural } d4.) c8
}

```



### *Legature di portamento con complesse strutture di tratteggio*

Le legature di portamento possono avere schemi di tratteggio complessi definendo la proprietà `dash-definition`. `dash-definition` è una lista di `dash-elements`. Un `dash-element` è una lista di parametri che definiscono il comportamento del tratteggio per un segmento della legatura.

La legatura di portamento è definita come il parametro `t` della curva di bezier che va da 0 sul margine sinistro della legatura fino a 1 su quello destro. `dash-element` è una lista di (`inizio-t fine-t frazione-trattino punto-trattino`). La regione della legatura di portamento che va da `inizio-t` a `fine-t` avrà una frazione `frazione-trattino` di ogni `punto-trattino` nero. `punto-trattino` viene definito in spazi rigo. `frazione-trattino` è impostato su 1 per una legatura di portamento continua.

```

\relative c' {
  \once \override
    Slur.dash-definition = #'((0 0.3 0.1 0.75)
                              (0.3 0.6 1 1)
                              (0.65 1.0 0.4 0.75))

  c4( d e f)
  \once \override
    Slur.dash-definition = #'((0 0.25 1 1)
                              (0.3 0.7 0.4 0.75)
                              (0.75 1.0 1 1))

  c4( d e f)
}

```



## Vedi anche

Glossario Musicale: Sezione “legatura di portamento” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Sul non annidamento di parentesi e legature di valore” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.4.2 [Direction and placement], pagina 611, `<undefined>` [Phrasing slurs], pagina `<undefined>`.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Slur” in *Guida al Funzionamento Interno*.



## Legature di frase

Le *legature di frase*, che indicano una frase musicale, si scrivono con i comandi `\(` e `\)`:

```
\relative {
  c''4\( d( e) f(
  e2) d\)
}
```



A livello tipografico, una legatura di frase si comporta in modo pressoché identico a una normale legatura di portamento. Sono però trattate come oggetti diversi; ad esempio, `\slurUp` non ha effetto su una legatura di frase. Le legature di frase possono essere posizionate sopra o sotto il rigo, come è spiegato in Sezione 5.4.2 [Direction and placement], pagina 611.

Per inserire più legature di frase simultanee o sovrapposte si usa `\=`, come per le normali legature di portamento (vedi `\(` [Slurs], pagina `\)`).

Le legature di frase possono essere continue, puntate o tratteggiate. Lo stile predefinito è quello continuo:

```
\relative {
  c'4\( e g2\)
  \phrasingSlurDashed
  g4\( e c2\)
  \phrasingSlurDotted
  c4\( e g2\)
  \phrasingSlurSolid
  g4\( e c2\)
}
```



Le legature di frase possono essere anche semitratteggiate (la prima metà tratteggiata, la seconda continua) o semicontinue (la prima metà continua, la seconda tratteggiata):

```
\relative {
  c'4\( e g2\)
  \phrasingSlurHalfDashed
  g4\( e c2\)
  \phrasingSlurHalfSolid
  c4\( e g2\)
  \phrasingSlurSolid
  g4\( e c2\)
}
```



Si possono definire modelli di tratteggio personalizzati anche per le legature di frase:

```
\relative {
```

```

c'4\ ( e g2\
\phrasingSlurDashPattern #0.7 #0.75
g4\ ( e c2\
\phrasingSlurDashPattern #0.5 #2.0
c4\ ( e g2\
\phrasingSlurSolid
g4\ ( e c2\
}

```



Le definizioni dei modelli di tratteggio per le legature di frase hanno la stessa struttura di quelle per le legature di portamento. Per maggiori informazioni sui modelli complessi di tratteggio si consultino i frammenti in [\[Slurs\]](#), pagina [\[Slurs\]](#).

## Comandi predefiniti

```

\phrasingSlurUp, \phrasingSlurDown, \phrasingSlurNeutral, \phrasingSlurDashed,
\phrasingSlurDotted, \phrasingSlurHalfDashed, \phrasingSlurHalfSolid,
\phrasingSlurDashPattern, \phrasingSlurSolid.

```

## Vedi anche

Manuale di apprendimento: Sezione “Sul non annidamento di parentesi e legature di valore” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.4.2 [Direction and placement], pagina 611, [\[Slurs\]](#), pagina [\[Slurs\]](#).

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “PhrasingSlur” in *Guida al Funzionamento Interno*.

## Respiri

I respiri si inseriscono col comando `\breathe`:

```
{ c''2. \breathe d''4 }
```



Diversamente da altri segni di espressione, il respiro non è associato alla nota precedente ma è un evento musicale separato. Dunque tutti i segni espressivi attaccati alla nota precedente, tutte le parentesi quadre che indicano le travature manuali e le parentesi che indicano le legature di portamento e di frase devono essere poste prima di `\breathe`.

Un respiro termina una travatura automatica; per evitare questo comportamento, si veda [\[Manual beams\]](#), pagina [\[Manual beams\]](#).

```
\relative { c''8 \breathe d e f g2 }
```



È supportata la divisio, indicatore del respiro nella musica antica. Maggiori dettagli in [\[Divisiones\]](#), pagina 444.

## Frammenti di codice selezionati

### *Cambiare il simbolo del segno di respiro*

Il glifo del respiro può essere modificato sovrascrivendo la proprietà `text` dell'oggetto di formattazione `BreathingSign` con qualsiasi testo incluso in un blocco markup.

```
\relative c'' {
  c2
  \override BreathingSign.text =
    \markup { \musicglyph #"scripts.rvarcomma" }
  \breathe
  d2
}
```



### *Usare un segno di spunta come simbolo di respiro*

La musica vocale e per fiati usa frequentemente il segno di spunta come segno di respiro. Questo indica un respiro che sottrae un po' di tempo alla nota precedente invece di prendere una piccola pausa, indicata dal segno di respiro rappresentato dalla virgola. Il segno può essere spostato un po' su per allontanarlo dal rigo.

```
\relative c'' {
  c2
  \breathe
  d2
  \override BreathingSign.Y-offset = #2.6
  \override BreathingSign.text =
    \markup { \musicglyph #"scripts.tickmark" }
  c2
  \breathe
  d2
}
```



### *Inserire una cesura*

I segni di cesura possono essere creati sovrascrivendo la proprietà `text` dell'oggetto `BreathingSign`. È disponibile anche un segno di cesura curvo.

```
\relative c'' {
  \override BreathingSign.text = \markup {
    \musicglyph #"scripts.caesura.straight"
  }
  c8 e4. \breathe g8. e16 c4

  \override BreathingSign.text = \markup {
    \musicglyph #"scripts.caesura.curved"
  }
  g8 e'4. \breathe g8. e16 c4
}
```

}



## Vedi anche

Glossario Musicale: Sezione “cesura” in *Glossario Musicale*.

Guida alla notazione: [Divisiones], pagina 444.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “BreathingEvent” in *Guida al Funzionamento Interno*, Sezione “BreathingSign” in *Guida al Funzionamento Interno*, Sezione “Breathing-sign-engraver” in *Guida al Funzionamento Interno*.

## Portamenti indeterminati discendenti (cadute) e ascendenti

I portamenti indeterminati verso il basso (cadute) e verso l’alto possono essere aggiunti alle note col comando `\bendAfter`. La direzione del portamento è indicata con un più o un meno (su o giù). Il numero indica l’intervallo per cui il portamento si estenderà *oltre* la nota principale.

```
\relative c' {
  c2\bendAfter #+4
  c2\bendAfter #-4
  c2\bendAfter #+6.5
  c2\bendAfter #-6.5
  c2\bendAfter #+8
  c2\bendAfter #-8
}
```



## Frammenti di codice selezionati

*Cambiare la forma dei portamenti indeterminati verso il basso o verso l’alto*

La proprietà `shortest-duration-space` può essere modificata per cambiare la forma dei portamenti indeterminati verso il basso o verso l’alto.

```
\relative c' {
  \override Score.SpacingSpanner.shortest-duration-space = #4.0
  c2-\bendAfter #5
  c2-\bendAfter #-4.75
  c2-\bendAfter #8.5
  c2-\bendAfter #-6
}
```



## Vedi anche

Glossario Musicale: Sezione “portamento indeterminato verso il basso” in *Glossario Musicale*,  
Sezione “portamento indeterminato verso l’alto” in *Glossario Musicale*.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

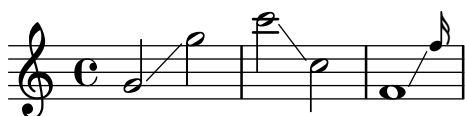
### 1.3.3 Indicazioni espressive lineari

Questa sezione spiega come creare varie indicazioni espressive che seguono una traiettoria lineare: glissandi, arpeggi e trilli.

#### Glissando

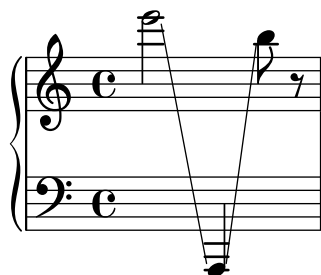
Un *glissando* si crea attaccando `\glissando` a una nota:

```
\relative {
  g'2\glissando g'
  c2\glissando c,
  \afterGrace f,1\glissando f'16
}
```



Un glissando può collegare note appartenenti a righe diversi:

```
\new PianoStaff <<
  \new Staff = "right" {
    e''2\glissando
    \change Staff = "left"
    a,,4\glissando
    \change Staff = "right"
    b''8 r |
  }
  \new Staff = "left" {
    \clef bass
    s1
  }
>>
```



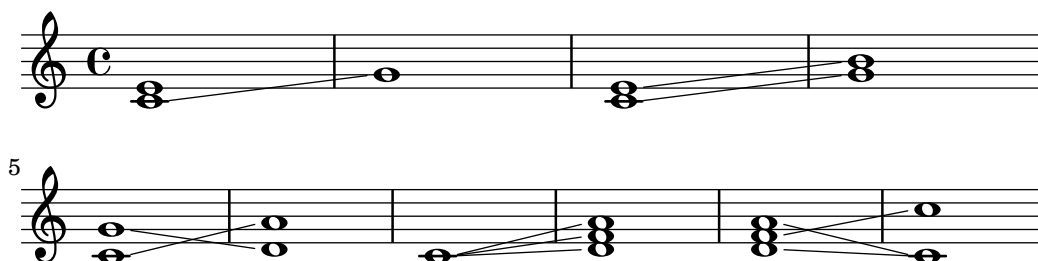
Un glissando può collegare le note negli accordi. Se è necessario qualcosa di diverso dal normale abbinamento uno a uno delle note, si possono definire le connessioni tra le note attraverso `\glissandoMap`, dove le note di un accordo sono numerate a partire da zero nell’ordine in cui appaiono nel file di input `.ly`.

```
\relative {
  <c' e>1\glissando g' |
```

```

<c, e>1\glissando |
<g' b> |
\break
\set glissandoMap = #'((0 . 1) (1 . 0))
<c, g'>1\glissando |
<d a'> |
\set glissandoMap = #'((0 . 0) (0 . 1) (0 . 2))
c1\glissando |
<d f a> |
\set glissandoMap = #'((2 . 0) (1 . 0) (0 . 1))
<f d a'>1\glissando |
<c c'> |
}

```



Si possono adottare diversi stili di glissando. Maggiori dettagli in Sezione 5.4.8 [Line styles], pagina 625.

## Frammenti di codice selezionati

### *Glissando contemporaneo*

Un glissando contemporaneo senza una nota finale può essere creato usando una nota nascosta e un tempo di cadenza.

```

\relative c'' {
  \time 3/4
  \override Glissando.style = #'zigzag
  c4 c
  \cadenzaOn
  c4\glissando
  \hideNotes
  c,,4
  \unHideNotes
  \cadenzaOff
  \bar "|"
}

```



### *Aggiungere i segni di tempo per i glissandi lunghi*

I battiti saltati nei glissandi molto lunghi vengono talvolta segnalati con delle indicazioni di tempo, che consistono solitamente in dei gambi privi di teste di nota. Questi gambi possono essere usati anche per contenere segni di espressione intermedi.

Se i gambi non si allineano bene al glissando, può essere necessario riposizionarli leggermente.

```
glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}
```

```
glissandoSkipOff = {
  \revert NoteColumn.glissando-skip
  \undo \hide NoteHead
  \revert NoteHead.no-ledgers
}
```

```
\relative c'' {
  r8 f8\glissando
  \glissandoSkipOn
  f4 g a a8\noBeam
  \glissandoSkipOff
  a8
```

```
  r8 f8\glissando
  \glissandoSkipOn
  g4 a8
  \glissandoSkipOff
  a8 |
```

```
  r4 f\glissando \<
  \glissandoSkipOn
  a4\f \>
  \glissandoSkipOff
  b8\! r |
}
```



*Lasciare che i glissandi vadano a capo*

Per permettere a un glissando di andare a capo se capita su un'interruzione di riga, si impostano le proprietà `breakable` e `after-line-breaking` su `##t`:

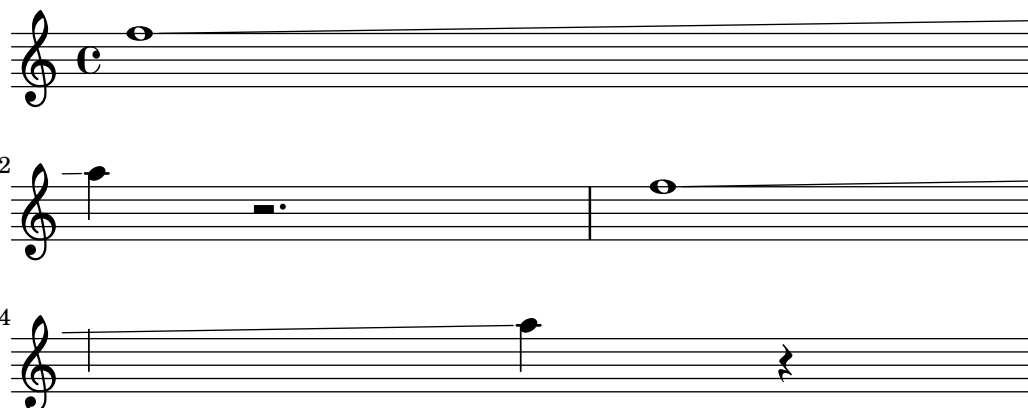
```
glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}
```

```
\relative c'' {
  \override Glissando.breakable = ##t
  \override Glissando.after-line-breaking = ##t
  f1\glissando |
  \break
}
```

```

a4 r2. |
f1\glissando
\once \glissandoSkipOn
\break
a2 a4 r4 |
}

```



#### *Estendere i glissandi sulle volte delle ripetizioni*

Un glissando che si estende in vari blocchi `\alternative` può essere simulato aggiungendo all'inizio di ogni blocco `\alternative` una nota di abbellimento nascosta da cui inizia un glissando. La nota di abbellimento deve avere la stessa altezza della nota da cui parte il glissando iniziale. In questo frammento si usa una funzione musicale che prende come argomento l'altezza della nota di abbellimento.

Attenzione: nella musica polifonica la nota di abbellimento deve avere una nota di abbellimento corrispondente in tutte le altre voci.

```

repeatGliss = #(define-music-function (grace)
  (ly:pitch?)
  #{
    % the next two lines ensure the glissando is long enough
    % to be visible
    \once \override Glissando.springs-and-rods
      = #ly:spanner::set-spacing-rods
    \once \override Glissando.minimum-length = #3.5
    \once \hideNotes
    \grace $grace \glissando
  #})

\score {
  \relative c'' {
    \repeat volta 3 { c4 d e f\glissando }
    \alternative {
      { g2 d }
      { \repeatGliss f g2 e }
      { \repeatGliss f e2 d }
    }
  }
}

music = \relative c' {

```



```

\voiceOne
\repeat volta 2 {
  g a b c\glissando
}
\alternative {
  { d1 }
  { \repeatGliss c \once \omit StringNumber e1\2 }
}
}

\score {
  \new StaffGroup <<
    \new Staff <<
      \context Voice { \clef "G_8" \music }
    >>
    \new TabStaff <<
      \context TabVoice { \clef "moderntab" \music }
    >>
  >>
}

```

## Vedi anche

Glossario Musicale: Sezione “glissando” in *Glossario Musicale*.

Guida alla notazione: Sezione 5.4.8 [Line styles], pagina 625.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Glissando” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Non è supportato il testo lungo la linea del glissando (ad esempio *gliss.*).

## Arpeggio

Un *arpeggio* su un accordo (detto anche accordo spezzato) si ottiene aggiungendo `\arpeggio` all'accordo:

```

\relative { <c' e g c>1\arpeggio }

```

Si possono scrivere vari tipi di arpeggio. `\arpeggioNormal` ripristina l'arpeggio normale:

```
\relative {
  <c' e g c>2\arpeggio

  \arpeggioArrowUp
  <c e g c>2\arpeggio

  \arpeggioArrowDown
  <c e g c>2\arpeggio

  \arpeggioNormal
  <c e g c>2\arpeggio
}
```



Si possono creare simboli di arpeggio speciali *in forma di parentesi*:

```
\relative {
  <c' e g c>2

  \arpeggioBracket
  <c e g c>2\arpeggio

  \arpeggioParenthesis
  <c e g c>2\arpeggio

  \arpeggioParenthesisDashed
  <c e g c>2\arpeggio

  \arpeggioNormal
  <c e g c>2\arpeggio
}
```



Le proprietà del tratteggio della parentesi dell'arpeggio sono regolate dalla proprietà `'dash-definition`, descritta in [\[Slurs\]](#), pagina [\[undefined\]](#).

Gli arpeggi possono essere scritti in modo esplicito con le legature di valore. Per maggiori dettagli si veda [\[Ties\]](#), pagina [\[undefined\]](#).

## Comandi predefiniti

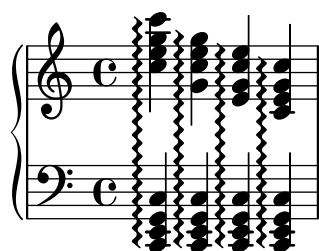
`\arpeggio`, `\arpeggioArrowUp`, `\arpeggioArrowDown`, `\arpeggioNormal`, `\arpeggioBracket`, `\arpeggioParenthesis` `\arpeggioParenthesisDashed`.

## Frammenti di codice selezionati

*Creare degli arpeggi che attraversano il rigo del pianoforte*

In un rigo per pianoforte (`PianoStaff`), è possibile far sì che un arpeggio attraversi i righi impostando la proprietà `PianoStaff.connectArpeggios = ##t`.

```
\new PianoStaff \relative c' ' <<
  \set PianoStaff.connectArpeggios = ##t
  \new Staff {
    <c e g c>4\arpeggio
    <g c e g>4\arpeggio
    <e g c e>4\arpeggio
    <c e g c>4\arpeggio
  }
  \new Staff {
    \clef bass
    \repeat unfold 4 {
      <c,, e g c>4\arpeggio
    }
  }
>>
```



*Creare degli arpeggi che attraversano i righi in altri contesti*

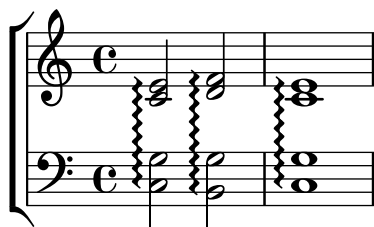
Si possono creare arpeggi che attraversano i righi in contesti diversi da `GrandStaff`, `PianoStaff` e `StaffGroup` se l'incisore `Span_arpeggio_engraver` è incluso nel contesto `Score`.

```
\score {
  \new ChoirStaff {
    \set Score.connectArpeggios = ##t
    <<
      \new Voice \relative c' {
        <c e>2\arpeggio
        <d f>2\arpeggio
        <c e>1\arpeggio
      }
      \new Voice \relative c {
        \clef bass
        <c g'>2\arpeggio
        <b g'>2\arpeggio
        <c g'>1\arpeggio
      }
    >>
  }
  \layout {
    \context {
      \Score
    }
  }
}
```

```

        \consists "Span_arpeggio_engraver"
    }
}
}

```



*Creare degli arpeggi che attraversano note appartenenti a voci diverse*

Si può disegnare un arpeggio che attraversa delle note in voci diverse dello stesso rigo se si aggiunge l'incisore `Span_arpeggio_engraver` nel contesto `Staff`:

```

\new Staff \with {
  \consists "Span_arpeggio_engraver"
}
\relative c' {
  \set Staff.connectArpeggios = ##t
  <<
    { <e' g>4\arpeggio <d f> <d f>2 }
    \\\
    { <d, f>2\arpeggio <g b>2 }
  >>
}

```



## Vedi anche

Glossario Musicale: Sezione “arpeggio” in *Glossario Musicale*.

Guida alla notazione: `<undefined>` [Slurs], pagina `<undefined>`, `<undefined>` [Ties], pagina `<undefined>`.

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Arpeggio” in *Guida al Funzionamento Interno*, Sezione “Slur” in *Guida al Funzionamento Interno*, Sezione “PianoStaff” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Non è possibile mostrare simultaneamente arpeggi connessi e non connessi in un `PianoStaff`.

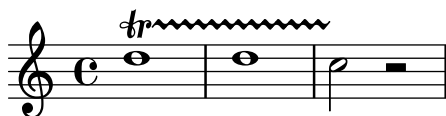
L'arpeggio in forma di parentesi non può essere impostato con facilità negli arpeggi che attraversano i righi; occorre ricorrere a metodi più complessi descritti in [Cross-staff stems], pagina 330.

## Trilli

I trilli senza linea di estensione si ottengono col comando `\trill`; si veda [\[Articulations and ornamentations\]](#), pagina [\[Articulations and ornamentations\]](#).

I trilli con linea di estensione si ottengono con `\startTrillSpan` e `\stopTrillSpan`:

```
\relative {
  d''1\startTrillSpan
  d1
  c2\stopTrillSpan
  r2
}
```



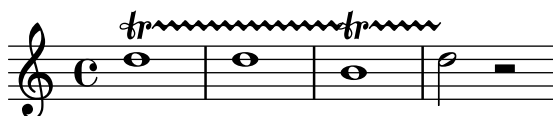
Un estensore del trillo che va a capo ricomincerà esattamente sopra la prima nota della nuova riga.

```
\relative {
  d''1\startTrillSpan
  \break
  d1
  c2\stopTrillSpan
  r2
}
```



È possibile tracciare trilli consecutivi senza dover esplicitare i comandi `\stopTrillSpan`, perché il trillo successivo diventerà automaticamente il limite destro di quello precedente.

```
\relative {
  d''1\startTrillSpan
  d1
  b1\startTrillSpan
  d2\stopTrillSpan
  r2
}
```



I trilli possono essere anche combinati con le note di abbellimento. La sintassi di questo costrutto e il metodo per posizionare in modo preciso gli abbellimenti sono descritti in [\[Grace notes\]](#), pagina [\[Grace notes\]](#).

```
\relative {
```

```

d''1~\afterGrace
d1\startTrillSpan { c32[ d]\stopTrillSpan }
c2 r2
}

```



I trilli che richiedono una nota ausiliaria dall'altezza esplicita si ottengono col comando `\pitchedTrill`. Il primo argomento è la nota principale e il secondo è la nota *trillata*, che appare come una testa di nota senza gambo e racchiusa tra parentesi.

```

\relative {
  \pitchedTrill
  d''2\startTrillSpan fis
  d2
  c2\stopTrillSpan
  r2
}

```



Alterazioni successive della stessa nota nella stessa misura devono essere aggiunte manualmente. Apparirà solo l'alterazione del primo trillo con notina in una misura.

```

\relative {
  \pitchedTrill
  eis''4\startTrillSpan fis
  eis4\stopTrillSpan
  \pitchedTrill
  eis4\startTrillSpan cis
  eis4\stopTrillSpan
  \pitchedTrill
  eis4\startTrillSpan fis
  eis4\stopTrillSpan
  \pitchedTrill
  eis4\startTrillSpan fis!
  eis4\stopTrillSpan
}

```



## Comandi predefiniti

`\startTrillSpan`, `\stopTrillSpan`.

## Vedi anche

Glossario Musicale: Sezione “trillo” in *Glossario Musicale*.

Guida alla notazione: [\[Articulations and ornamentations\]](#), pagina [\[Grace notes\]](#), pagina [\[Grace notes\]](#).

Frammenti: Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TrillSpanner” in *Guida al Funzionamento Interno*.

## 1.4 Ripetizioni



La ripetizione è un concetto chiave in musica e può essere resa con varie forme di notazione. LilyPond supporta i seguenti tipi di ripetizioni:

- volta**      La musica ripetuta non viene scritta per intero ma racchiusa tra barre di ripetizione. Se la ripetizione si trova all’inizio di un brano, la stanghetta di ritornello è posta soltanto alla fine della ripetizione. I finali alternativi (volte) appaiono da sinistra a destra e sono evidenziati da delle parentesi. Questa è la notazione standard per le ripetizioni con finali alternativi.
- unfold**    La musica ripetuta viene scritta per intero, tante volte quante sono specificate dal *numero-ripetizioni*. È utile quando si scrive musica ripetitiva.
- percent**   Si tratta di ripetizioni del singolo tempo (battito) o della battuta. Hanno l’aspetto di una barra obliqua o di segni di percentuale.
- tremolo**    Si usa per scrivere travature a tremolo.

### 1.4.1 Ripetizioni lunghe

Questa sezione spiega come inserire ripetizioni lunghe (solitamente di più battute). Tali ripetizioni possono essere in due forme: racchiuse tra segni di ritornello oppure ricopiate interamente (adatte a scrivere musica ripetitiva). Si possono anche controllare manualmente i segni di ripetizione.

## Ripetizioni normali

La sintassi per una normale ripetizione è

```
\repeat volta numero-di-ripetizioni espressione-musicale
```

dove *espressione-musicale* è la musica da ripetere.

Un'unica ripetizione senza finale alternativo:

```
\relative {
  \repeat volta 2 { c''4 d e f }
  c2 d
  \repeat volta 2 { d4 e f g }
}
```



Il segno di inizio della ripetizione, per impostazione predefinita, non appare nella prima misura. È tuttavia possibile aggiungerlo inserendo a mano la battuta `\bar ".|:"` prima della prima nota.

```
\relative {
  \repeat volta 2 { \bar ".|:" c''4 d e f }
  c2 d
  \repeat volta 2 { d4 e f g }
}
```



I finali alternativi si ottengono con `\alternative`. Ogni gruppo di alternative deve essere a sua volta racchiuso tra parentesi.

```
\repeat volta numero-di-ripetizioni espressione-musicale
\alternative {
  { espressione-musicale }
}
```

dove *espressione-musicale* è la musica.

Se il numero di ripetizioni è superiore a quello dei finali alternativi, alle prime ripetizioni viene assegnata la prima alternativa.

Una singola ripetizione con un finale alternativo:

```
\relative {
  \repeat volta 2 { c''4 d e f | }
  \alternative {
    { c2 e | }
    { f2 g | }
  }
  c1
}
```





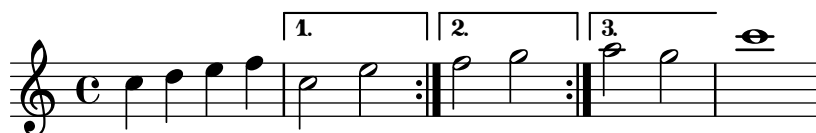
Molteplici ripetizioni con un finale alternativo:

```
\relative {
  \repeat volta 4 { c''4 d e f | }
  \alternative {
    { c2 e | }
    { f2 g | }
  }
  c1
}
```



Molteplici ripetizioni con più di un finale alternativo:

```
\relative {
  \repeat volta 3 { c''4 d e f | }
  \alternative {
    { c2 e | }
    { f2 g | }
    { a2 g | }
  }
  c1
}
```



**Nota:** Se ci sono due o più finali alternativi, non ci deve essere niente tra la parentesi di chiusura di uno e quella di apertura di quello successivo all'interno di un blocco `\alternative`, altrimenti non si otterrà il numero atteso di finali.

**Nota:** Se si usa `\relative` dentro a un blocco `\repeat` senza istanziare esplicitamente il contesto `Voice`, appare un rigo in più (non desiderato). Vedi Sezione “Appare un rigo in più” in *Uso del Programma*.

Se una ripetizione che non ha finali alternativi inizia in mezzo a una misura, solitamente termina in un punto corrispondente nel mezzo di una misura successiva (così che tra le due estremità ci sia una misura completa). In questo caso i segni di ripetizione non sono delle “vere” e proprie stanghette, dunque né i controlli di battuta né i comandi `\partial` devono essere messi lì:

```
\relative {
  c'4 e g
  \repeat volta 4 {
    e4 |
    c2 e |
    g4 g g
  }
```

```

}
g4 |
a2 a |
g1 |
}

```



Se una ripetizione senza finali alternativi inizia con una misura parziale, si applicano gli stessi principi dell'esempio precedente, a parte il fatto che è richiesto un comando `\partial` all'inizio della misura:

```

\partial 4
\repeat volta 4 {
  e'4 |
  c2 e |
  g4 g g
}
g4 |
a2 a |
g1 |

```

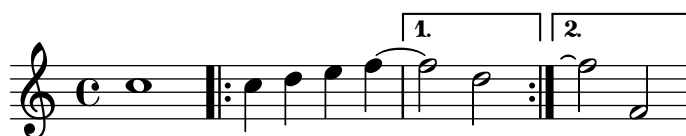


Si possono aggiungere delle legature di valore a un secondo finale:

```

\relative {
  c''1
  \repeat volta 2 { c4 d e f~ }
  \alternative {
    { f2 d }
    { f2\repeatTie f, }
  }
}

```



Il comando `\inStaffSegno` può essere usato per generare una stanghetta composita che incorpora il simbolo di segno nella stanghetta di ripetizione appropriata se usato col comando `\repeat volta`. Il tipo corretto di stanghetta di ripetizione, ovvero inizio della ripetizione, fine della ripetizione e doppia ripetizione, viene selezionato automaticamente. Il corrispondente segno “D.S.” deve essere aggiunto manualmente.

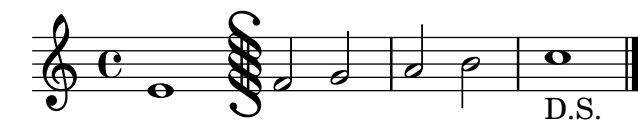
Lontano da una ripetizione:

```

\relative {
  e'1
  \inStaffSegno
  f2 g a b
}

```

```
c1_"D.S." \bar "|"."
```



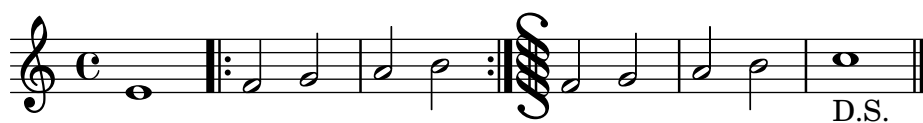
All'inizio di una ripetizione:

```
\relative {
  e'1
  \repeat volta 2 {
    \inStaffSegno % start repeat
    f2 g a b
  }
  c1_"D.S." \bar "|"."
```



Alla fine di una ripetizione:

```
\relative {
  e'1
  \repeat volta 2 {
    f2 g a b
    \inStaffSegno % end repeat
  }
  f2 g a b
  c1_"D.S." \bar "|"."
```



Tra due ripetizioni:

```
\relative {
  e'1
  \repeat volta 2 {
    f2 g a b
  }
  \inStaffSegno % double repeat
  \repeat volta 2 {
    f2 g a b
  }
  c1_"D.S." \bar "|"."
```



Si possono impostare simboli alternativi delle stanghette modificando nel contesto `Score` le proprietà `segnoType`, `startRepeatSegnoType`, `endRepeatSegnoType` o `doubleRepeatSegnoType` per il tipo di stanghetta richiesto. I tipi di stanghetta alternativi devono essere selezionati dai tipi predefiniti o dai tipi precedentemente definiti col comando `\defineBarLine` (vedi `<undefined>` [Bar lines], pagina `<undefined>`).

```
\defineBarLine ":"|.S[" #'(":". "S[" "]")
\defineBarLine "]" #'("]" "" """)
\relative {
  e'1
  \repeat volta 2 {
    f2 g a b
    \once \set Score.endRepeatSegnoType = ":"|.S["
    \inStaffSegno
  }
  f2 g \bar "]" a b
  c1_"D.S." \bar "|."
}
```



## Frammenti di codice selezionati

*Accorciare le parentesi delle volte*

Per impostazione predefinita, le parentesi delle volte si estendono per tutta l'alternativa, ma si possono accorciare impostando `voltaSpannerDuration`. Nell'esempio seguente, la parentesi dura una misura, che ha una durata di 3/4.

```
\relative c'' {
  \time 3/4
  c4 c c
  \set Score.voltaSpannerDuration = #(ly:make-moment 3/4)
  \repeat volta 5 { d4 d d }
  \alternative {
    {
      e4 e e
      f4 f f
    }
    { g4 g g }
  }
}
```



*Aggiungere le parentesi delle volte a altri righi*

L'incisore `Volta_engraver` risiede nel contesto `Score`, quindi le parentesi delle ripetizioni appaiono di norma soltanto sul rigo superiore. Questo comportamento può essere modificato

aggiungendo l'incisore `Volta_engraver` al contesto `Staff` in cui si desidera far apparire le parentesi; si veda anche il frammento “Volta multirigo”.

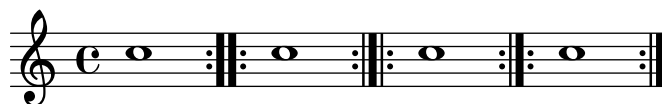
```
<<
\new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
\new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
\new Staff \with { \consists "Volta_engraver" } { c'2 g' e' a' }
\new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
>>
```



#### *Impostare la doppia ripetizione predefinita per le volte*

Esistono tre diversi stili di doppie ripetizioni per le volte, che si possono impostare con `doubleRepeatType`.

```
\relative c' {
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":...:"
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":|.|:"
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":|..:"
  \repeat volta 1 { c1 }
}
```



#### *Numeri di battuta alternativi*

Si possono impostare due metodi alternativi di numerazione della battuta, utili specialmente per le ripetizioni.

```
\relative c'{
  \set Score.alternativeNumberingStyle = #'numbers
  \repeat volta 3 { c4 d e f | }
  \alternative {
    { c4 d e f | c2 d \break }
    { f4 g a b | f4 g a b | f2 a | \break }
    { c4 d e f | c2 d }
  }
```

```

    }
c1 \break
\set Score.alternativeNumberingStyle = #'numbers-with-letters
\repeat volta 3 { c,4 d e f | }
    \alternative {
        { c4 d e f | c2 d \break }
        { f4 g a b | f4 g a b | f2 a | \break }
        { c4 d e f | c2 d }
    }
c1
}

```

The musical score consists of six staves, each representing a different alternative of a repeated musical phrase. The first staff is labeled '1.' and shows a melody in treble clef with a repeat sign. The second staff is labeled '2' and shows a different melody in treble clef with a repeat sign. The third staff is labeled '2' and shows a third melody in treble clef with a repeat sign. The fourth staff is labeled '5' and shows a fourth melody in treble clef with a repeat sign. The fifth staff is labeled '6b' and shows a fifth melody in treble clef with a repeat sign. The sixth staff is labeled '6c' and shows a sixth melody in treble clef with a repeat sign.

## Vedi anche

Glossario Musicale: Sezione “ripetizione” in *Glossario Musicale*, Sezione “volta” in *Glossario Musicale*.

Guida alla notazione: [\[Bar lines\]](#), pagina [\[Modifying context plug-ins\]](#), pagina 587, [\[Modifying ties and slurs\]](#), pagina 632, [\[Time administration\]](#), pagina [\[Time administration\]](#).

File installati: `ly/engraver-init.ly`.

Frammenti: Sezione “Repeats” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “VoltaBracket” in *Guida al Funzionamento Interno*, Sezione “RepeatedMusic” in *Guida al Funzionamento Interno*, Sezione “VoltaRepeatedMusic” in *Guida al Funzionamento Interno*, Sezione “UnfoldedRepeatedMusic” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Le legature di portamento che si estendono da un blocco `\repeat` verso un blocco `\alternative` funzioneranno solo nel primo finale alternativo. L'aspetto grafico di una legatura di portamento che continua negli altri finali alternativi può essere simulato con `\repeatTie` se la legatura si estende solo su una nota del blocco dell'alternativa, sebbene questo metodo non funzioni in `TabStaff`. Altri metodi che si possono adattare per indicare legature di portamento che continuino su varie note dei blocchi di alternativa, e che funzionano anche nei contesti `TabStaff`, sono presentati in [Modifying ties and slurs], pagina 632.

Inoltre le legature di portamento non possono ricollegarsi dalla fine di un'alternativa all'inizio della ripetizione.

I glissandi che si estendono da un blocco `\repeat` in un blocco `\alternative` funzioneranno soltanto per il primo finale alternativo. L'aspetto grafico di un glissando che continua negli altri finali alternativi può essere indicato creando un glissando che inizia su una nota di abbellimento nascosta. Vedere ad esempio il frammento “Estendere i glissandi attraverso le ripetizioni” nei Frammenti Selezionati in [Glissando], pagina 140.

Se una ripetizione che inizia con una misura incompleta ha un blocco `\alternative` che contiene modifiche alla proprietà `measureLength`, l'uso di `\unfoldRepeats` causerà l'erroneo posizionamento delle stanghette e degli avvisi di controllo di battuta.

Una ripetizione annidata come la seguente

```
\repeat ...
\repeat ...
\alternative
```

è ambigua, perché non è chiaro a quale `\repeat` appartenga il blocco `\alternative`. Questa ambiguità si risolve facendo in modo che `\alternative` appartenga sempre al blocco `\repeat` interno. Per chiarezza, si consiglia di usare le parentesi in queste situazioni.

## Indicazioni di ripetizione manuali

**Nota:** Questi metodi vengono usati solo per mostrare tipi di ripetizioni inusuali, e potrebbero causare un comportamento inaspettato. Nella maggior parte dei casi, le ripetizioni devono essere create col comando standard `\repeat` oppure stampando le stanghette opportune. Maggiori informazioni in `<undefined>` [Bar lines], pagina `<undefined>`.

La proprietà `repeatCommands` permette di controllare la formattazione delle ripetizioni. Il suo valore è una lista Scheme dei comandi di ripetizione.

**start-repeat**

Stampa una stanghetta `.|:.`

```
\relative {
  c'1
  \set Score.repeatCommands = #'(start-repeat)
  d4 e f g
  c1
}
```



Come vuole la pratica comune di incisione, i segni di ripetizione non vengono stampati all'inizio di un brano.

`end-repeat`

Stampa una stanghetta :|.:.

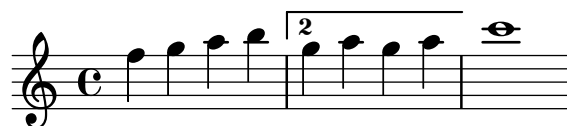
```
\relative {
  c''1
  d4 e f g
  \set Score.repeatCommands = #'(end-repeat)
  c1
}
```



`(volta numero) ... (volta #f)`

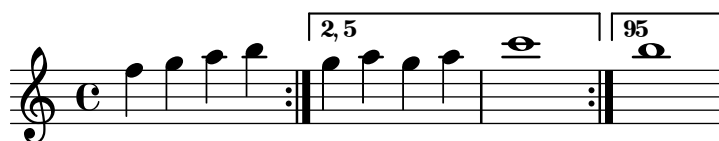
Crea una nuova volta col numero specificato. La parentesi della volta deve essere terminata esplicitamente, altrimenti non sarà stampata.

```
\relative {
  f''4 g a b
  \set Score.repeatCommands = #'((volta "2"))
  g4 a g a
  \set Score.repeatCommands = #'((volta #f))
  c1
}
```



Comandi di ripetizione multipli possono trovarsi nello stesso punto:

```
\relative {
  f''4 g a b
  \set Score.repeatCommands = #'((volta "2, 5") end-repeat)
  g4 a g a
  c1
  \set Score.repeatCommands = #'((volta #f) (volta "95") end-repeat)
  b1
  \set Score.repeatCommands = #'((volta #f))
}
```



Si può includere del testo nella parentesi della volta. Il testo può consistere di un numero, di più numeri o di un'indicazione testuale, si veda `\markup { 1. 2. 3... \text \italic { ad lib. } }` [Formatting text], pagina `\markup { 1. 2. 3... \text \italic { ad lib. } }`. Il modo più semplice per usare del testo è definirlo prima e poi includerlo nella lista Scheme,

```
voltaAdLib = \markup { 1. 2. 3... \text \italic { ad lib. } }
```



```

\relative {
  c''1
  \set Score.repeatCommands =
    #(list(list 'volta voltaAdLib) 'start-repeat)
  c4 b d e
  \set Score.repeatCommands = #'((volta #f) (volta "4.") end-repeat)
  f1
  \set Score.repeatCommands = #'((volta #f))
}

```



## Vedi anche

Guida alla notazione: [\[Bar lines\]](#), pagina [\[Formatting text\]](#), pagina [\[Formatting text\]](#).

Frammenti: Sezione “Repeats” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “VoltaBracket” in *Guida al Funzionamento Interno*, Sezione “RepeatedMusic” in *Guida al Funzionamento Interno*, Sezione “VoltaRepeatedMusic” in *Guida al Funzionamento Interno*.

## Ripetizioni ricopiate

Col comando `unfold`, le ripetizioni possono servire a semplificare la scrittura di musica ripetitiva. La sintassi è

```
\repeat unfold numero-di-ripetizioni espressione-musicale
```

dove *espressione-musicale* è la musica e *numero-di-ripetizioni* è il numero di volte per cui è ripetuta *espressione-musicale*.

```

\relative {
  \repeat unfold 2 { c''4 d e f }
  c1
}

```



In alcuni casi, specialmente in un contesto `\relative`, la funzione `\repeat unfold` non equivale a riscrivere l'espressione musicale più volte. Ad esempio

```
\repeat unfold 2 { a'4 b c }
```

non equivale a

```
a'4 b c | a'4 b c
```

Le ripetizioni dispiegate (`unfold`) possono avere dei finali alternativi.

```

\relative {
  \repeat unfold 2 { c''4 d e f }
  \alternative {
    { c2 g' }
    { c,2 b }
  }
}

```

```

}
c1
}

```



Se il numero di ripetizioni è maggiore del numero di finali alternativi, la prima alternativa viene applicata più volte, finché le alternative rimaste non esauriscono il numero totale delle ripetizioni.

```

\relative {
  \repeat unfold 4 { c''4 d e f }
  \alternative {
    { c2 g' }
    { c,2 b }
    { e2 d }
  }
  c1
}

```



Se il numero di finali alternativi è maggiore del numero di ripetizioni, solo le prime alternative vengono applicate. Le alternative rimanenti saranno ignorate e non verranno stampate.

```

\relative {
  \repeat unfold 2 { c''4 d e f }
  \alternative {
    { c2 g' }
    { c,2 b }
    { e2 d }
  }
  c1
}

```



È anche possibile annidare molteplici funzioni `unfold` (con o senza finali alternativi).

```

\relative {
  \repeat unfold 2 {
    \repeat unfold 2 { c''4 d e f }
    \alternative {
      { c2 g' }
      { c,2 b }
    }
  }
  c1
}

```



Gli accordi si ripetono col simbolo di ripetizione dell'accordo `q`. Vedi [\[Chord repetition\]](#), pagina [\[Chord repetition\]](#).

**Nota:** Se si usa `\relative` dentro a un blocco `\repeat` senza istanziare esplicitamente il contesto `Voice`, appare un rigo in più (non desiderato). Vedi Sezione “Appare un rigo in più” in *Uso del Programma*.

## Vedi anche

Guida alla notazione: [\[Chord repetition\]](#), pagina [\[Chord repetition\]](#).

Frammenti: Sezione “Repeats” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “RepeatedMusic” in *Guida al Funzionamento Interno*, Sezione “UnfoldedRepeatedMusic” in *Guida al Funzionamento Interno*.

### 1.4.2 Ripetizioni brevi

Questa sezione tratta il modo in cui inserire brevi ripetizioni. Le ripetizioni brevi possono avere due forme: segni di tratto obliquo o percentuale per rappresentare le ripetizioni di una singola nota, di una singola misura o di due misure; tremolo negli altri casi.

#### Ripetizioni con percentuale

Brevi sezioni ripetute vengono stampate la prima volta e le ripetizioni vengono sostituite da un apposito segno.

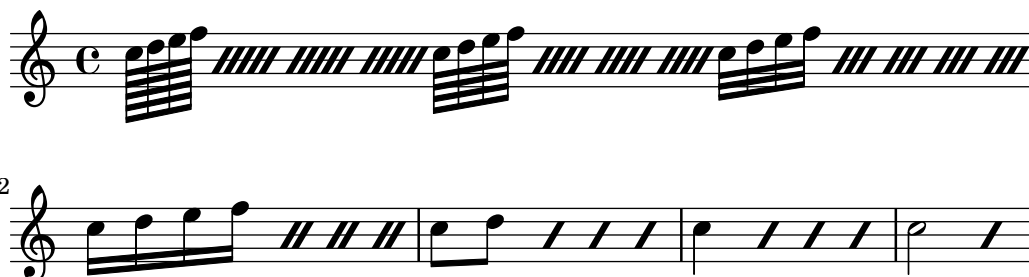
La sintassi è

```
\repeat percent numero espressione-musicale
```

dove *espressione-musicale* è l'espressione musicale da ripetere.

Fraseggi più brevi di una misura vengono sostituiti dal tratto obliquo.

```
\relative c'' {
  \repeat percent 4 { c128 d e f }
  \repeat percent 4 { c64 d e f }
  \repeat percent 5 { c32 d e f }
  \repeat percent 4 { c16 d e f }
  \repeat percent 4 { c8 d }
  \repeat percent 4 { c4 }
  \repeat percent 2 { c2 }
}
```



Fraseggi di una o due misure vengono sostituiti da simboli simili alla percentuale.

```
\relative c'' {
  \repeat percent 2 { c4 d e f }
```

```
\repeat percent 2 { c2 d }
\repeat percent 2 { c1 }
}
```

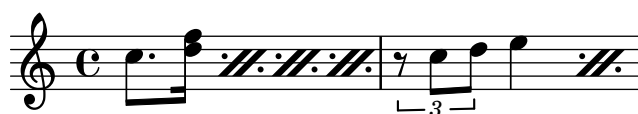


```
\relative {
  \repeat percent 3 { c''4 d e f | c2 g' }
}
```



Fraseggi più brevi di una misura ma con durate miste adottano un simbolo di doppia percentuale.

```
\relative {
  \repeat percent 4 { c''8. <d f>16 }
  \repeat percent 2 { \tuplet 3/2 { r8 c d } e4 }
}
```



## Frammenti di codice selezionati

### *Contatore della ripetizione con segno percentuale*

Le ripetizioni di misura che hanno più di due ripetizioni possono avere un contatore se si cambia la proprietà opportuna, come mostra questo esempio:

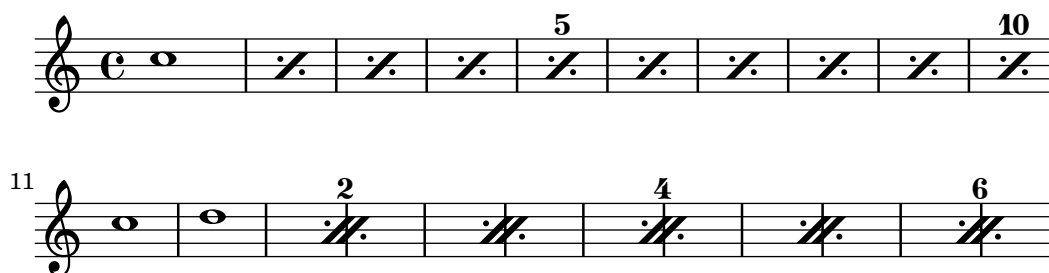
```
\relative c'' {
  \set countPercentRepeats = ##t
  \repeat percent 4 { c1 }
}
```



### *Visibilità del conto della ripetizione con segno percentuale*

I contatori della ripetizione con segno percentuale possono essere mostrati a intervalli regolari impostando la proprietà di contesto `repeatCountVisibility`.

```
\relative c'' {
  \set countPercentRepeats = ##t
  \set repeatCountVisibility = #(every-nth-repeat-count-visible 5)
  \repeat percent 10 { c1 } \break
  \set repeatCountVisibility = #(every-nth-repeat-count-visible 2)
  \repeat percent 6 { c1 d1 }
}
```



### Ripetizioni con segni di percentuale isolati

Si possono stampare anche segni di percentuale isolati.

```
makePercent =
#(define-music-function (note) (ly:music?)
  "Make a percent repeat the same length as NOTE."
  (make-music 'PercentEvent
    'length (ly:music-length note)))

\relative c'' {
  \makePercent s1
}
```



Vedi anche

Glossario Musicale: Sezione “percent repeat” in *Glossario Musicale*, Sezione “simile” in *Glossario Musicale*.

Frammenti: Sezione “Repeats” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “RepeatSlash” in *Guida al Funzionamento Interno*, Sezione “RepeatSlashEvent” in *Guida al Funzionamento Interno*, Sezione “DoubleRepeatSlash” in *Guida al Funzionamento Interno*, Sezione “PercentRepeat” in *Guida al Funzionamento Interno*, Sezione “PercentRepeatCounter” in *Guida al Funzionamento Interno*, Sezione “PercentRepeatedMusic” in *Guida al Funzionamento Interno*, Sezione “Percent\_repeat\_engraver” in *Guida al Funzionamento Interno*, Sezione “DoublePercentEvent” in *Guida al Funzionamento Interno*, Sezione “DoublePercentRepeat” in *Guida al Funzionamento Interno*, Sezione “DoublePercentRepeatCounter” in *Guida al Funzionamento Interno*, Sezione “Double\_percent\_repeat\_engraver” in *Guida al Funzionamento Interno*, Sezione “Slash\_repeat\_engraver” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

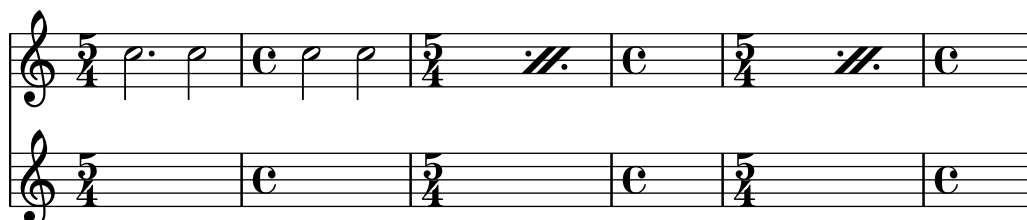
Le ripetizioni con percentuale non contengono nient'altro che il segno di percentuale; in particolare, i cambi di tempo non saranno ripetuti.

```
\repeat percent 3 { \time 5/4 c2. 2 \time 4/4 2 2 }
```



Qualsiasi cambio di tempo o comando `\partial` devono trovarsi in passaggi paralleli *esterni* a qualsiasi ripetizione con percentuale, per esempio su una traccia di tempo separata.

```
<<
  \repeat percent 3 { c2. 2 2 2 }
  \repeat unfold 3 { \time 5/4 s4*5 \time 4/4 s1 }
>>
```

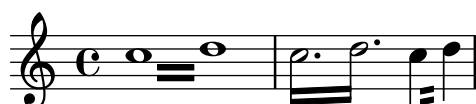


## Ripetizioni con tremolo

I tremoli possono avere due forme: alternanza tra due note, o due accordi, e rapida ripetizione di una singola nota o accordo. I tremoli costituiti da un'alternanza si indicano con delle travature che collegano le note o gli accordi che si alternano, mentre i tremoli che consistono in una rapida ripetizione di una nota singola si indicano aggiungendo delle travature o dei tratti di suddivisione obliqui alla singola nota.

Per inserire i segni del tremolo tra le note, si usa `\repeat` con lo stile tremolo:

```
\relative c'' {
  \repeat tremolo 8 { c16 d }
  \repeat tremolo 6 { c16 d }
  \repeat tremolo 2 { c16 d }
}
```



La sintassi di `\repeat tremolo` prevede specificamente che all'interno delle parentesi siano indicate due note, e che il numero di ripetizioni corrisponda a un valore espresso in durate di note normali o puntate. Dunque `\repeat tremolo 7` è valido e produce una nota doppiamente puntata, mentre `\repeat tremolo 9` non è valido.

La durata del tremolo equivale alla durata dell'espressione musicale tra parentesi moltiplicata per il numero di ripetizioni: `\repeat tremolo 8 { c16 d16 }` corrisponde a un tremolo di una semibreve, rappresentata come due semibrevi unite dalle travature del tremolo.

Ci sono due modi di inserire dei segni di tremolo su una singola nota. Anche in questo caso si usa la sintassi `\repeat tremolo`, ma la nota non deve essere racchiusa tra parentesi:

```
\repeat tremolo 4 c'16
```



Si può ottenere lo stesso output aggiungendo : $N$  dopo la nota, dove  $N$  indica la durata della suddivisione (deve essere almeno 8). Se  $N$  è 8, viene aggiunta una travatura al gambo della nota. Se  $N$  è omissso, viene usato l'ultimo valore:

```
\relative {
```

```

c' '2:8 c:32
c: c:
}

```



## Frammenti di codice selezionati

### *Tremoli attraverso i righi*

Dato che `\repeat tremolo` si aspetta esattamente due argomenti musicali per i tremoli di accordi, la nota o l'accordo che cambiano rigo in un tremolo che attraversa i righi devono essere posti tra parentesi graffe insieme al comando `\change Staff`.

```

\new PianoStaff <<
  \new Staff = "up" \relative c' ' {
    \key a \major
    \time 3/8
    s4.
  }
  \new Staff = "down" \relative c' ' {
    \key a \major
    \time 3/8
    \voiceOne
    \repeat tremolo 6 {
      <a e'>32
      {
        \change Staff = "up"
        \voiceTwo
        <cis a' dis>32
      }
    }
  }
}
>>

```



## Vedi anche

Frammenti: Sezione “Repeats” in *Frammenti di codice*.

## 1.5 Note simultanee

The image displays three staves of musical notation, likely for a piano. The first staff shows a sequence of chords with dynamics *f*, *p*, *tr*, and *pp*. The second staff, starting at measure 112, continues the sequence with *tr* and *pp*. The third staff, starting at measure 116, shows a transition from *p* to *f*. The notation includes various note values, rests, and dynamic markings.

La polifonia in musica si riferisce alla coesistenza simultanea di più di una voce in un brano musicale. In LilyPond la polifonia si riferisce alla coesistenza di più voci sullo stesso rigo.

### 1.5.1 Una voce

In questa sezione vengono spiegate le note simultanee che fanno parte di un'unica voce.

#### Note in un accordo

Un accordo si forma racchiudendo una serie di altezze tra `<` e `>` e può essere seguito da una durata, come accade per le semplici note.

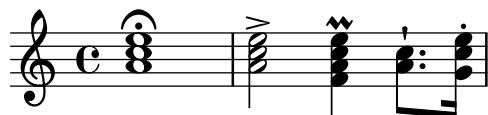
```
\relative {
  <a' c e>1 <a c e>2 <f a c e>4 <a c>8. <g c e>16
}
```

The image shows a musical staff with a common time signature (C). It contains a sequence of chords: a triad of A, C, and E; a triad of A, C, and E; a triad of F, A, and C; and a triad of A, C, and E. The notation uses vertical lines to group the notes of each chord.



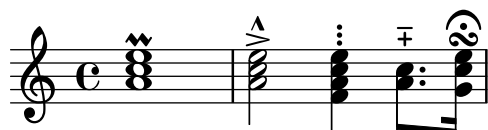
Proprio come per le note, si possono specificare le articolazioni da riferire all'accordo.

```
\relative {
  <a' c e>1\fermata <a c e>2-> <f a c e>4\prall <a c>8.^! <g c e>16-.
}
```



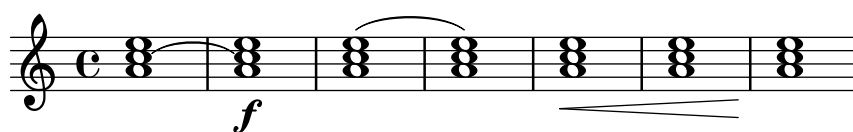
Si possono specificare abbellimenti e articolazioni per ogni nota che fa parte dell'accordo.

```
\relative {
  <a' c>\prall e>1 <a-> c-^ e>2 <f-. a c-. e-.>4
  <a-+ c-->8. <g\fermata c e\turn>16
}
```



Tuttavia, alcuni elementi della notazione, come le dinamiche e le forcelle, devono essere attaccate all'accordo invece che alle sue singole note, altrimenti non appariranno. Altri elementi della notazione, come le diteggiature e le legature di portamento, saranno posizionate in modo nettamente diverso se attaccate alle note di un accordo invece che a un accordo intero o a singole note.

```
\relative {
  <a'\f c( e>1 <a c) e>\f <a\< c e>( <a\! c e>)
  <a c e>\< <a c e> <a c e>\!
}
```



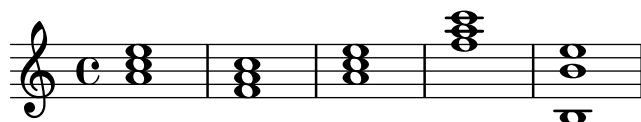
Un accordo si comporta semplicemente come un contenitore di note, articolazioni e altri elementi. Di conseguenza, un accordo privo di note non ha una durata. Qualsiasi articolazione attaccata a un accordo vuoto si troverà nel momento musicale della nota o accordo seguenti e si combinerà con questi (possibilità più complesse di combinazione sono spiegate in [\[Simultaneous expressions\]](#), pagina [\(undefined\)](#)):

```
\relative {
  \grace { g'8( a b }
  <> ) \p \< -. -\markup \italic "sempre staccato"
  \repeat unfold 4 { c4 e } c1\f
}
```



Si può usare la modalità relativa per indicare l'altezza degli accordi. La prima nota di ogni accordo è sempre relativa alla prima nota dell'accordo che lo precede oppure, se non c'è un accordo precedente, è relativa all'altezza dell'ultima nota che precede l'accordo. Le altezze di tutte le altre note dell'accordo sono relative alla nota che le precede *all'interno dell'accordo*.

```
\relative {
  <a' c e>1 <f a c> <a c e> <f' a c> <b, e b,>
}
```



Maggiori informazioni sugli accordi si trovano in Sezione 2.7 [Chord notation], pagina 411.

## Vedi anche

Glossario Musicale: Sezione “accordo” in *Glossario Musicale*.

Manuale d'apprendimento: Sezione “Combinare le note negli accordi” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 2.7 [Chord notation], pagina 411, [\[Articulations and ornamentations\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [Relative octave entry], pagina [\[undefined\]](#), [\[undefined\]](#) [Multiple voices], pagina [\[undefined\]](#).

Frammenti: Sezione “Simultaneous notes” in *Frammenti di codice*.

## Problemi noti e avvertimenti

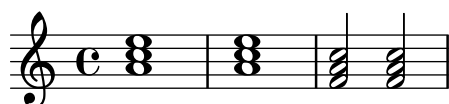
Gli accordi che contengono più di due altezze in uno spazio del rigo, come ad esempio ‘<e f! fis!>’, presentano le teste di tali note sovrapposte. A seconda della situazione, si può migliorare l'aspetto con

- l'uso temporaneo di [\[undefined\]](#) [Multiple voices], pagina [\[undefined\]](#), ‘<< f! \\[\[undefined\]](#) <e fis!> >>’,
- la trascrizione enarmonica di una o più altezze, ‘<e f ges>’, oppure
- i [\[undefined\]](#) [Clusters], pagina [\[undefined\]](#).

## Ripetizione di un accordo

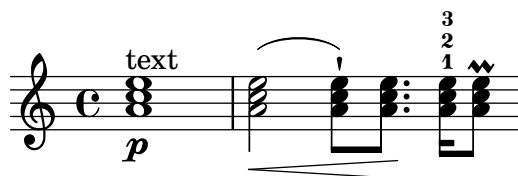
Per inserire la musica più rapidamente, si può usare una scorciatoia che ripete l'accordo precedente. Il simbolo di ripetizione dell'accordo è q:

```
\relative {
  <a' c e>1 q <f a c>2 q
}
```



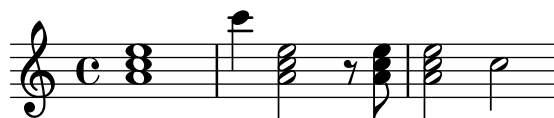
Come nel caso dei normali accordi, il simbolo di ripetizione dell'accordo si può usare con le durate, le articolazioni, i testi a margine, le legature di portamento, le travature, etc. dato che solo le altezze dell'accordo precedente vengono duplicate.

```
\relative {
  <a' c e>1\p~"text" q2\<( q8)[-! q8.]\\! q16-1-2-3 q8\prall
}
```



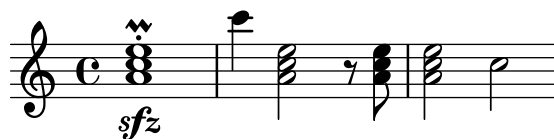
Il simbolo di ripetizione dell'accordo ricorda sempre l'ultimo accordo inserito, quindi è possibile inserire l'accordo più recente anche se nel frattempo sono state inserite altre note (senza accordi) o pause.

```
\relative {
  <a' c e>1 c'4 q2 r8 q8 |
  q2 c, |
}
```



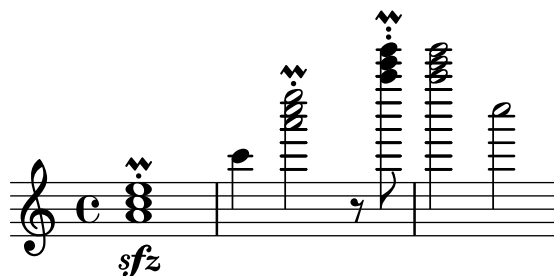
Tuttavia questo simbolo non conserva le dinamiche, le articolazioni o gli abbellimenti dell'accordo precedente.

```
\relative {
  <a'-. c\prall e>1\sffz c'4 q2 r8 q8 |
  q2 c, |
}
```



Per far sì che alcuni elementi siano conservati, si può invocare esplicitamente la funzione `\chordRepeats` con un'ulteriore argomento che indica una lista di *tipi di evento* da mantenere, a meno che eventi di quel tipo non siano già presenti nell'accordo `q` stesso.

```
\relative {
  \chordRepeats #'(articulation-event)
  { <a'-. c\prall e>1\sffz c'4 q2 r8 q8-. } |
  q2 c, |
}
```

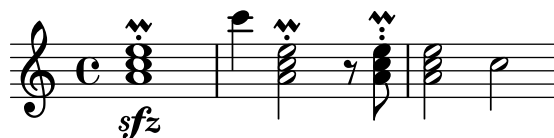


In questo esempio l'uso di `\chordRepeats` all'interno di un blocco `\relative` produce risultati indesiderati: gli eventi di un accordo, una volta espansi, non si distinguono più per essere stati inseriti come accordi normali, quindi `\relative` assegna un'ottava basata sul contesto corrente.

Dato che `\relative` annidati non si influenzano l'un l'altro, si può usare un altro `\relative` dentro `\chordRepeats` per stabilire le relazioni di ottava prima di espandere gli accordi ripetuti.

In questo caso l'intero contenuto del `\relative` più interno non influenza quello esterno; ecco perché in questo esempio la nota finale è stata specificata con un'ottava diversa.

```
\new Voice
\relative c'' {
  \chordRepeats #'(articulation-event)
  \relative
  { <a'- . c\prall e>1\s fz c'4 q2 r8 q8- . } |
  q2 c |
}
```



Le interazioni con `\relative` si verificano solo con chiamate esplicite di `\chordRepeats`: l'espansione implicita all'inizio della creazione della partitura viene fatta in un momento in cui tutti i `\relative` sono stati già elaborati.

## Vedi anche

Guida alla notazione: Sezione 2.7 [Chord notation], pagina 411, `<undefined>` [Articulations and ornamentations], pagina `<undefined>`.

File installati: `ly/chord-repetition-init.ly`.

## Espressioni simultanee

Una o più espressioni musicali racchiuse tra due coppie di parentesi uncinate sono considerate simultanee. Se la prima espressione inizia con una nota singola o se l'intera espressione simultanea appare esplicitamente all'interno di una voce, sarà posta in un solo rigo; altrimenti gli elementi dell'espressione simultanea saranno messi in righe separate.

Gli esempi seguenti mostrano espressioni simultanee su un rigo:

```
\new Voice { % voce singola esplicita
  << \relative { a'4 b g2 }
    \relative { d'4 g c,2 } >>
}
```



```
\relative {
  % prima nota singola
  a' << \relative { a'4 b g }
    \relative { d'4 g c, } >>
}
```



Questo può essere utile se le sezioni simultanee hanno durate identiche, ma i tentativi di collegare note con durate diverse allo stesso gambo causerà degli errori. Le note, le articolazioni

e le modifiche delle proprietà in una *singola* voce ('Voice') sono raccolte e create secondo l'ordine della musica:

```
\relative {
  <a' c>4-. <>-. << c a >> << { c-. <c a> } { a s-. } >>
}
```



Per poter inserire gambi o travature multiple e variare le durate o altre proprietà di note riferibili allo stesso momento musicale, occorre usare più voci.

L'esempio seguente mostra come le espressioni simultanee possano generare implicitamente righe multipli:

```
% nessuna singola nota precede l'espressione simultanea
<< \relative { a'4 b g2 }
  \relative { d'4 g2 c,4 } >>
```



In questo caso le durate diverse non causano problemi perché sono interpretate in voci diverse.

## Problemi noti e avvertimenti

Se le note appartenenti a due o più voci, senza che sia stato specificato uno spostamento, hanno i gambi nella stessa direzione, il messaggio

attenzione: questa voce ha bisogno di un'impostazione `\voiceXx` o `\shiftXx` apparirà durante la compilazione del file. Si può evitare con:

```
\override NoteColumn.ignore-collision = ##t
```

Tuttavia, questo comando non si limita a eliminare l'avvertimento, ma impedisce qualsiasi risoluzione delle collisioni, e potrebbe comportare altri effetti indesiderati (vedi anche i *Problemi noti* in `<undefined>` [Collision resolution], pagina `<undefined>`).

## Cluster

Un cluster prescrive l'esecuzione simultanea di tutti i suoni compresi in un determinato intervallo. Può essere rappresentato come un involucro che contiene le note che ne fanno parte. Si inserisce applicando la funzione `\makeClusters` a una sequenza di accordi, ad esempio:

```
\relative \makeClusters { <g' b>2 <c g'> }
```



Si possono inserire insieme sullo stesso rigo le normali note e i cluster, anche contemporaneamente. In tal caso non viene fatto alcun tentativo di evitare automaticamente collisioni tra le note normali e i cluster.

## Vedi anche

Glossario Musicale: Sezione “cluster” in *Glossario Musicale*.

Frammenti: Sezione “Simultaneous notes” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “ClusterSpanner” in *Guida al Funzionamento Interno*, Sezione “ClusterSpannerBeacon” in *Guida al Funzionamento Interno*, Sezione “Cluster\_spanner\_engraver” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

I cluster hanno un buon aspetto solo se durano almeno per due accordi; altrimenti appaiono troppo stretti.

I cluster non hanno un gambo e non possono indicare delle durate da soli, ma la lunghezza del cluster è determinata dalle durate degli accordi che lo definiscono. Più cluster distinti devono essere separati da una pausa.

I cluster non generano output MIDI.

### 1.5.2 Più voci

Questa sezione presenta le note simultanee in più voci o più righi.

## Polifonia su un solo rigo

### *Istanziare esplicitamente le voci*

La struttura di base necessaria per ottenere più voci indipendenti in un solo rigo è illustrata nell'esempio seguente:

```
\new Staff <<
  \new Voice = "prima"
    \relative { \voiceOne r8 r16 g' e8. f16 g8[ c,] f e16 d }
  \new Voice= "seconda"
    \relative { \voiceTwo d'16 c d8~ 16 b c8~ 16 b c8~ 16 b8. }
>>
```



Le voci sono istanziate esplicitamente e vengono contrassegnate da dei nomi. I comandi `\voiceOne ... \voiceFour` impostano le voci in modo che la prima e terza voce abbiano i gambi in su, la seconda e la quarta voce i gambi in giù, le teste di nota della terza e quarta voce siano spostate orizzontalmente e le pause in ciascuna voce siano spostate automaticamente per evitare collisioni. Il comando `\oneVoice` ripristina tutte le impostazioni della voce alle direzioni neutrali predefinite.

### *Passaggi polifonici temporanei*

Un passaggio polifonico temporaneo si può creare col seguente costrutto:

```
<< { \voiceOne ... }
  \new Voice { \voiceTwo ... }
>> \oneVoice
```

In questo esempio la prima espressione all'interno di un passaggio polifonico temporaneo è posta nel contesto `Voice` che era in uso immediatamente prima del passaggio polifonico e quello stesso contesto `Voice` continua dopo la sezione temporanea. Le altre espressioni comprese nelle

parentesi uncinate vengono assegnate a voci temporanee distinte. Questo permette di assegnare il testo a una voce che continua prima, durante e dopo una sezione polifonica:

```
\relative <<
  \new Voice = "melody" {
    a'4
    <<
      {
        \voiceOne
        g f
      }
      \new Voice {
        \voiceTwo
        d2
      }
    >>
    \oneVoice
    e4
  }
  \new Lyrics \lyricsto "melody" {
    This is my song.
  }
>>
```



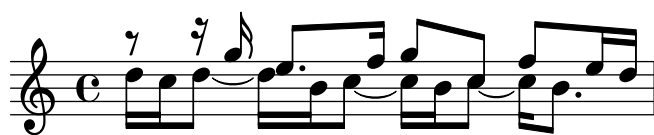
I comandi `\voiceOne` e `\voiceTwo` sono necessari per definire le impostazioni di ogni voce.

### *Il costrutto con la doppia barra inversa (backslash)*

Il costrutto `<< {...} \ {...} >>`, in cui due (o più) espressioni sono separate da due barre inverse (backslash), si comporta diversamente dal costrutto simile privo delle due barre: *tutte* le espressioni in questo costrutto vengono assegnate a nuovi contesti `Voice`. Questi nuovi contesti `Voice` vengono creati implicitamente e ad essi vengono assegnati dei nomi prestabiliti "1", "2", etc.

Il primo esempio potrebbe essere riscritto nel modo seguente:

```
<<
  \relative { r8 r16 g'' e8. f16 g8[ c,] f e16 d }
  \
  \relative { d''16 c d8~ 16 b c8~ 16 b c8~ 16 b8. }
>>
```



Questa sintassi si usa quando non importa che le voci temporanee siano create e poi eliminate. A queste voci create implicitamente vengono assegnate le impostazioni equivalenti all'uso dei comandi `\voiceOne ... \voiceFour`, nell'ordine in cui appaiono nell'input.

Nell'esempio seguente, la voce intermedia ha i gambi in su, dunque viene inserita in terza posizione in modo che diventi la terza voce, che ha i gambi in su. Si usano le pause spaziatrici per evitare il raddoppio delle pause ordinarie.

```
<<
\relative { r8 g'' g g g f16 ees f8 d }
\\
\relative { ees'8 r ees r d r d r }
\\
\relative { d''8 s c s bes s a s }
>>
```



In tutti i brani, a eccezione di quelli più semplici, è consigliabile creare contesti **Voice** espliciti come è spiegato in Sezione “Contesti e incisori” in *Manuale di Apprendimento* e Sezione “Definire esplicitamente le voci” in *Manuale di Apprendimento*.

### Ordine delle voci

Quando si inseriscono più voci nel file di input, conviene usare il seguente ordine:

```
Voce 1: la più alta
Voce 2: la più bassa
Voce 3: la seconda più alta
Voce 4: la seconda più bassa
Voce 5: la terza più alta
Voce 6: la terza più bassa
etc.
```

Sebbene possa sembrare controintuitivo, ciò semplifica il processo automatico di formattazione. Si noti che le voci con numero dispari hanno i gambi in su, quelle con numero pari hanno i gambi in giù:

```
\new Staff <<
\time 2/4
{ f''2 } % 1: la più alta
\\
{ c'2 } % 2: la più bassa
\\
{ d''2 } % 3: seconda più alta
\\
{ e'2 } % 4: seconda più bassa
\\
{ b'2 } % 5: terza più alta
\\
{ g'2 } % 6: terza più bassa
>>
```





**Nota:** Il testo e gli estensori (come le legature di portamento e di valore, le forcelle, etc.) non possono ‘attraversare’ le voci.

### ***Durate identiche***

Nel caso speciale in cui si desideri inserire sezioni musicali parallele con il medesimo ritmo, si possono combinare in un unico contesto **Voice**, formando dunque degli accordi. Per farlo, vanno racchiuse in un semplice costrutto musicale simultaneo all’interno di una voce esplicita:

```
\new Voice <<
  \relative { e''4 f8 d e16 f g8 d4 }
  \relative { c''4 d8 b c16 d e8 b4 }
>>
```



Questo metodo produce strane travature e avvertimenti se le sezioni musicali non hanno lo stesso ritmo.

### **Comandi predefiniti**

`\voiceOne`, `\voiceTwo`, `\voiceThree`, `\voiceFour`, `\oneVoice`.

### **Vedi anche**

Manuale d’apprendimento: Sezione “Le voci contengono la musica” in *Manuale di Apprendimento*, Sezione “Definire esplicitamente le voci” in *Manuale di Apprendimento*.

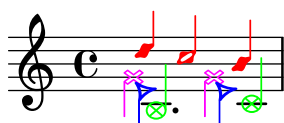
Guida alla notazione: [Percussion staves], pagina 390, `\invisible` [Invisible rests], pagina `\invisible`, `\invisible` [Stems], pagina `\invisible`.

Frammenti: Sezione “Simultaneous notes” in *Frammenti di codice*.

### **Stili di voce**

Si possono assegnare colori e forme diverse a ciascuna voce per facilitarne l’identificazione:

```
<<
  \relative { \voiceOneStyle d''4 c2 b4 }
  \\\
  \relative { \voiceTwoStyle e'2 e }
  \\\
  \relative { \voiceThreeStyle b2. c4 }
  \\\
  \relative { \voiceFourStyle g'2 g }
>>
```



Il comando `\voiceNeutralStyle` permette di ripristinare l’aspetto predefinito.

### **Comandi predefiniti**

`\voiceOneStyle`, `\voiceTwoStyle`, `\voiceThreeStyle`, `\voiceFourStyle`, `\voiceNeutralStyle`.

## Vedi anche

Manuale d'apprendimento: Sezione “Sento le Voci” in *Manuale di Apprendimento*, Sezione “Altre fonti di informazione” in *Manuale di Apprendimento*.

Frammenti: Sezione “Simultaneous notes” in *Frammenti di codice*.

## Risoluzione delle collisioni

Le teste di note che si trovano in voci diverse ma hanno stessa altezza, stessa testa e direzione del gambo opposta vengono unite automaticamente; invece, le note che hanno la stessa testa o la stessa direzione del gambo non vengono unite. Le pause opposte a un gambo in una voce diversa vengono spostate verticalmente. L'esempio seguente mostra tre diverse circostanze, sul primo e terzo movimento della prima battuta e sul primo movimento della seconda battuta, in cui l'unione automatica delle teste di nota non funziona.

```
<<
  \relative {
    c''8 d e d c d c4
    g'2 fis
  } \\\
  \relative {
    c''2 c8. b16 c4
    e,2 r
  } \\\
  \relative {
    \oneVoice
    s1
    e'8 a b c d2
  }
>>
```



Note con teste diverse possono essere unite come è mostrato sotto. In questo esempio le teste delle note nel primo battito della prima battuta sono unite:

```
<<
  \relative {
    \mergeDifferentlyHeadedOn
    c''8 d e d c d c4
    g'2 fis
  } \\\
  \relative {
    c''2 c8. b16 c4
    e,2 r
  } \\\
  \relative {
    \oneVoice
    s1
    e'8 a b c d2
  }
>>
```



Le minime e le semiminime, invece, non sono unite, perché sarebbe difficile distinguerle.

Anche le teste di note con diversi punti, come nel terzo movimento della prima battuta, possono essere unite:

```
<<
\relative {
  \mergeDifferentlyHeadedOn
  \mergeDifferentlyDottedOn
  c''8 d e d c d c4
  g'2 fis
} \\\
\relative {
  c''2 c8. b16 c4
  e,2 r
} \\\
\relative {
  \oneVoice
  s1
  e'8 a b c d2
}
>>
```



La minima e la croma all'inizio della seconda misura sono unite per errore, perché l'unione automatica non riesce a completare correttamente l'unione quando tre o più note sono allineate sulla stessa colonna di note: in questo caso la testa di nota unita non è corretta. Per far sì che l'unione selezioni la testa di nota corretta, occorre applicare il comando `\shiftOn` alla nota che non deve essere unita. In questo esempio si usa `\shiftOn` per spostare il Sol superiore (g) fuori dalla colonna e di conseguenza `\mergeDifferentlyHeadedOn` funziona correttamente.

```
<<
\relative {
  \mergeDifferentlyHeadedOn
  \mergeDifferentlyDottedOn
  c''8 d e d c d c4
  \shiftOn
  g'2 fis
} \\\
\relative {
  c''2 c8. b16 c4
  e,2 r
} \\\
\relative {
  \oneVoice
  s1
  e'8 a b c d2
}
>>
```



Il comando `\shiftOn` permette (senza forzare) lo spostamento delle note in una voce. Quando si applica `\shiftOn` a una voce, una nota o accordo in quella voce vengono spostati solo se il suo gambo altrimenti entrerebbe in collisione col gambo di un'altra voce, e solo se i gambi che collidono puntano nella stessa direzione. Il comando `\shiftOff` impedisce che si verifichi questo tipo di spostamento.

Per impostazione predefinita, le voci più esterne (solitamente la prima e la seconda voce) hanno specificato `\shiftOff`, mentre le voci più interne (terza e successive) hanno specificato `\shiftOn`. Quando si applica uno spostamento, le voci con i gambi in su (voci dispari) vengono spostate a destra, e le voci con i gambi in giù (voci pari) vengono spostate a sinistra.

Ecco un esempio che aiuta a visualizzare come un'espressione simultanea abbreviata viene espansa internamente.

**Nota:** Attenzione: con tre o più voci, l'ordine verticale delle voci nel file di input non deve essere lo stesso dell'ordine verticale delle voci del rigo!

```
\new Staff \relative {
  %% inserimento abbreviato
  <<
    { f'2 } % 1: più alta
    \\\
    { g,2 } % 2: più bassa
    \\\
    { d'2 } % 3: più alta centrale
    \\\
    { b2 } % 4: più bassa centrale
  >>
  %% espansione interna dell'input precedente
  <<
    \new Voice = "1" { \voiceOne \shiftOff f'2 }
    \new Voice = "2" { \voiceTwo \shiftOff g,2 }
    \new Voice = "3" { \voiceThree \shiftOn d'2 } % sposta a destra
    \new Voice = "4" { \voiceFour \shiftOn b2 } % sposta a sinistra
  >>
}
```



Due ulteriori comandi, `\shiftOnn` e `\shiftOnnn`, mettono a disposizione altri livelli di spostamento che possono essere specificati in modo temporaneo per risolvere delle collisioni in situazioni complesse – vedi Sezione “Esempio musicale” in *Manuale di Apprendimento*.

Le note vengono unite solo se presentano opposta direzione dei gambi (come accade, ad esempio, nella prima o seconda voce o quando i gambi sono impostati esplicitamente in direzioni opposte).

## Comandi predefiniti

`\mergeDifferentlyDottedOn`, `\mergeDifferentlyDottedOff`, `\mergeDifferentlyHeadedOn`, `\mergeDifferentlyHeadedOff`.

```
\shiftOn, \shiftOnn, \shiftOnnn, \shiftOff.
```

## Frammenti di codice selezionati

*Voci ulteriori per evitare le collisioni*

In alcuni casi di musica polifonica complessa sono necessarie delle voci ulteriori per evitare le collisioni tra note. Se servono più di quattro voci parallele, si possono aggiungere altre voci definendo una variabile con la funzione Scheme function `context-spec-music`.

```
voiceFive = #(context-spec-music (make-voice-props-set 4) 'Voice)
```

```
\relative c' {
  \time 3/4
  \key d \minor
  \partial 2
  <<
    \new Voice {
      \voiceOne
      a4. a8
      e'4 e4. e8
      f4 d4. c8
    }
    \new Voice {
      \voiceTwo
      d,2
      d4 cis2
      d4 bes2
    }
    \new Voice {
      \voiceThree
      f'2
      bes4 a2
      a4 s2
    }
    \new Voice {
      \voiceFive
      s2
      g4 g2
      f4 f2
    }
  >>
}
```



*Forzare lo spostamento orizzontale delle note*

Quando il motore tipografico non riesce a risolvere una situazione, si può usare la sintassi che sovrascrive le decisioni tipografiche. L'unità di misura usata è lo spazio del rigo.

```
\relative c' <<
{
```

```

    <d g>2 <d g>
  }
  \\
  {
    <b f'>2
    \once \override NoteColumn.force-hshift = #1.7
    <b f'>2
  }
>>

```



## Vedi anche

Glossario Musicale: Sezione “polifonia” in *Glossario Musicale*.

Manuale d'apprendimento: Sezione “Note simultanee” in *Manuale di Apprendimento*, Sezione “Le voci contengono la musica” in *Manuale di Apprendimento*, Sezione “Esempio musicale” in *Manuale di Apprendimento*.

Frammenti: Sezione “Simultaneous notes” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “NoteColumn” in *Guida al Funzionamento Interno*, Sezione “NoteCollision” in *Guida al Funzionamento Interno*, Sezione “RestCollision” in *Guida al Funzionamento Interno*.

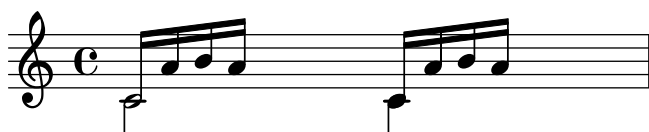
## Problemi noti e avvertimenti

Se si usa `\override NoteColumn.ignore-collision = ##t`, le note con teste diverse che si trovano in voci diverse saranno unite in modo non corretto.

```

\mergeDifferentlyHeadedOn
<< \relative { c'16 a' b a } \\ \relative { c'2 } >>
\override NoteColumn.ignore-collision = ##t
<< \relative { c'16 a' b a } \\ \relative { c'2 } >>

```



## Combinazione automatica delle parti

La combinazione automatica delle parti si usa per combinare in un unico rigo due parti musicali separate. Ciò è utile soprattutto quando si scrivono partiture orchestrali. Viene stampata una sola voce se le due parti musicali sono identiche, ma nei punti in cui sono diverse viene aggiunta una seconda voce. Le direzioni dei gambi sono impostate in su e in giù in base alla voce di appartenenza, mentre le sezioni solistiche e *a due* sono a loro volta identificate e contrassegnate.

La sintassi per la combinazione automatica delle parti è:

```

\partcombine espressione-musicale1 espressione-musicale2

```

L'esempio seguente illustra il funzionamento di base: le parti sono poste su un unico rigo in modo polifonico e le direzioni dei gambi sono regolate di conseguenza. Si usano le stesse variabili per le parti indipendenti e il rigo combinato.

```

instrumentOne = \relative {

```

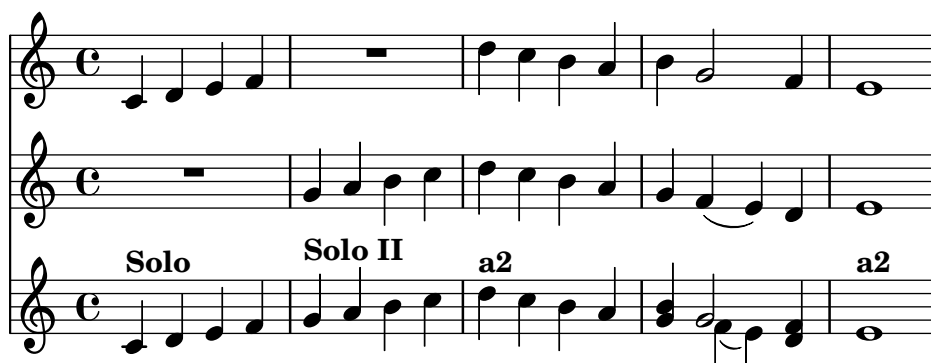
```

    c'4 d e f |
    R1 |
    d'4 c b a |
    b4 g2 f4 |
    e1 |
  }

instrumentTwo = \relative {
  R1 |
  g'4 a b c |
  d4 c b a |
  g4 f( e) d |
  e1 |
}

<<
  \new Staff \instrumentOne
  \new Staff \instrumentTwo
  \new Staff \partcombine \instrumentOne \instrumentTwo
>>

```



Entrambe le parti hanno note identiche nella terza misura, dunque viene stampata una sola nota. Le direzioni dei gambi e delle legature di portamento e di valore sono impostate automaticamente, a seconda che l'esecuzione delle parti sia solistica o all'unisono. Quando si rende necessario, in caso di polifonia, la prima parte (nel contesto *one*) ha i gambi in “su”, mentre la seconda (nel contesto *two*) ha sempre i gambi in “giù”. In caso di parti solistiche, la prima e seconda parte sono contrassegnate rispettivamente con “Solo” e “Solo II”. Le parti (*a due*) all'unisono sono contrassegnate con la scritta “a2”.

Il comportamento predefinito del combinatore delle parti è quello di unire due note della stessa altezza in una nota *a due*, combinare in un accordo note della stessa durata e separate da un intervallo inferiore alla nona e separare in voci distinte le note con un intervallo superiore alla nona (o quando le voci collidono). Tale comportamento può essere modificato con un argomento opzionale di una coppia di numeri dopo il comando `\partcombine`: il primo indica l'intervallo in cui le note iniziano a essere combinate (il valore predefinito è zero) e il secondo dove le note vengono divise in note distinte. Impostando il secondo argomento su zero, il combinatore delle parti divide le note che hanno un intervallo di una seconda o più; impostandolo su uno, divide le note di una terza o più, e così via.

```

instrumentOne = \relative {
  a4 b c d |
  e f g a |
}

```

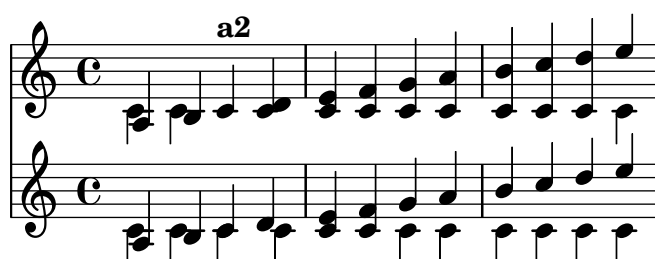
```

    b c d e |
  }

instrumentTwo = \relative {
  c'4 c c c |
  c c c c |
  c c c c |
}

<<
  \new Staff \partcombine \instrumentOne \instrumentTwo
  \new Staff \partcombine #'(2 . 3) \instrumentOne \instrumentTwo
>>

```



Entrambi gli argomenti di `\partcombine` sono interpretati come contesti `Voice` separati, dunque se la musica viene inserita in modo relativo *entrambe* le parti devono contenere una funzione `\relative`, ovvero:

```

\partcombine
  \relative ... espressione-musicale1
  \relative ... espressione-musicale2

```

Un blocco `\relative` che racchiude un `\partcombine` non ha effetto sulle altezze di `espressione-musicale1` e `espressione-musicale2`.

Nelle partiture professionali, spesso le voci sono tenute separate per lunghi passaggi anche se alcune note sono le stesse in entrambe le voci e potrebbero essere stampate come unisono. Combinare le note in un accordo o mostrare una voce come solista, dunque, non è la scelta ideale, perché la funzione `\partcombine` considera ogni nota individualmente. In questo caso si può sovrascrivere la funzione `\partcombine` con uno dei comandi elencati sotto. Tutti i comandi devono essere preceduti da `\once` per applicarli soltanto alla nota successiva dell'espressione musicale.

- `\partcombineApart` mantiene le note su due voci distinte, anche se potrebbero essere combinate in un accordo o in un unisono.
- `\partcombineChords` unisce le note in un accordo.
- `\partcombineUnisono` unisce entrambe le voci come “unisono”.
- `\partcombineSoloI` stampa soltanto la prima voce e la contrassegna come un “Solo”.
- `\partcombineSoloII` stampa soltanto la seconda voce e la contrassegna come un “Solo”.
- `\partcombineAutomatic` termina le funzioni dei comandi precedenti e ripristina il funzionamento predefinito di `\partcombine`.

```

instrumentOne = \relative c' {
  \partcombineApart c2^"separato" e |
  \partcombineAutomatic e2^"automatico" e |
  \partcombineChords e'2^"accordo" e |
  \partcombineAutomatic c2^"automatico" c |
}

```



```

\partcombineApart c2^"separato" \once \partcombineChords e^"accordo una volta sola"
c2 c |
}
instrumentTwo = \relative {
  c'2 c |
  e2 e |
  a,2 c |
  c2 c' |
  c2 c |
  c2 c |
}

<<
  \new Staff { \instrumentOne }
  \new Staff { \instrumentTwo }
  \new Staff { \partcombine \instrumentOne \instrumentTwo }
>>

```

The image displays three staves of musical notation, each with a treble clef and a common time signature (C). The notes are c2, e2, a2, c2, c2, c2. Above the staves, labels indicate the settings used for each staff: 'separato automatico' for the top staff, 'automatico a2' for the middle staff, and 'separato a2' for the bottom staff. The notes are grouped into pairs, and the labels 'accordo' and 'accordo una volta sola' are placed above the notes in the top and bottom staves respectively.

### Uso di `\partcombine` col testo vocale

Il comando `\partcombine` non è progettato per funzionare col testo vocale; al punto che se una delle voci è nominata in modo esplicito per poterle assegnare del testo, l'unione delle parti smette di funzionare. Tuttavia, questo risultato si può ottenere usando un contesto `NullVoice`. Vedi [\[Polyphony with shared lyrics\]](#), pagina [\[Polyphony with shared lyrics\]](#).

### Frammenti di codice selezionati

#### *Combinare due parti sullo stesso rigo*

Lo strumento di unione delle parti (il comando `\partcombine`) permette di combinare varie parti sullo stesso rigo. Indicazioni testuali come “solo” e “a2” sono aggiunte automaticamente; per toglierle basta impostare la proprietà `printPartCombineTexts` su `f`. Per le partiture vocali (inni), non c'è bisogno di aggiungere i testi “solo/a2”, quindi dovrebbero essere disattivati. Tuttavia potrebbe convenire non usarlo se c'è una qualche parte solista, perché non verrebbe indicata. In tali casi è preferibile usare la notazione polifonica normale.

Questo frammento illustra i tre modi con cui due parti possono essere stampate su uno stesso rigo: normale polifonia, `\partcombine` senza testo e `\partcombine` con testo.

```
%% Combining pedal notes with clef changes
```

```
musicUp = \relative c' {
```

```

\time 4/4
a4 c4.( g8) a4 |
g4 e' g,( a8 b) |
c b a2.
}

musicDown = \relative c'' {
  g4 e4.( d8) c4 |
  r2 g'4( f8 e) |
  d2 \stemDown a
}

\score {
  <<
    <<
      \new Staff {
        \set Staff.instrumentName = #"Standard polyphony"
        << \musicUp \\\musicDown >>
      }
      \new Staff \with { printPartCombineTexts = ##f } {
        \set Staff.instrumentName = #"PartCombine without texts"
        \partcombine \musicUp \musicDown
      }
      \new Staff {
        \set Staff.instrumentName = #"PartCombine with texts"
        \partcombine \musicUp \musicDown
      }
    >>
  >>
  \layout {
    indent = 6.0\cm
    \context {
      \Score
      \override SystemStartBar.collapse-height = #30
    }
  }
}

```

Standard polyphony

PartCombine without texts

PartCombine with texts



*Modificare le indicazioni testuali di partcombine*

Quando si usa la funzionalità di combinazione automatica delle parti, si può modificare il testo delle sezioni soliste e dell'unisone:

```
\new Staff <<
  \set Staff.soloText = #"girl"
  \set Staff.soloIIIText = #"boy"
  \set Staff.aDueText = #"together"
  \partcombine
    \relative c'' {
      g4 g r r
      a2 g
    }
    \relative c'' {
      r4 r a( b)
      a2 g
    }
  >>
```

**Vedi anche**

Glossario Musicale: Sezione “a due” in *Glossario Musicale*, Sezione “parte” in *Glossario Musicale*.

Guida alla notazione: `<undefined>` [Writing parts], pagina `<undefined>`.

Frammenti: Sezione “Simultaneous notes” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “PartCombineMusic” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*.

**Problemi noti e avvertimenti**

Tutte le funzioni `\partcombine...` possono accettare soltanto due voci.

Le funzioni `\partcombine...` non possono essere inserite all’interno di un blocco `\tuplet` o `\relative`.

Se `\printPartCombineTexts` è attivo e le due voci eseguono le stesse note “in modo discontinuo” nella stessa misura, potrebbe apparire il testo `a2` più di una volta in quella misura.

`\partcombine` sa soltanto quando una nota inizia in una voce (Voice); non può, ad esempio, ricordare se una nota in una voce è già iniziata quando combina le note già iniziate nell’altra voce. Questo può portare a vari problemi inattesi, tra cui la stampa non corretta dei segni “Solo” e “Unisone”.

`\partcombine` tiene tutti gli estensori (legature di portamento e di valore, forcelle, etc.) nella stessa voce, quindi se uno di questi estensori inizia o termina in una voce diversa potrebbe essere stampato incorrettamente o non essere stampato affatto.

Se la funzione `\partcombine` non riesce a combinare le due espressioni musicali (ovvero quando le due voci hanno durate diverse), assegnerà alle voci, internamente, nomi personalizzati: rispettivamente `one` e `two`. Ciò significa che se c’è un “passaggio” a un contesto Voice nominato diversamente, gli eventi in quel contesto verranno ignorati.

Consultare i *Problemi noti e avvertimenti* in [Default tablatures], pagina 341, se si usa `\partcombine` con l'intavolatura, e la *Nota* in `\undefined` [Automatic beams], pagina `\undefined`, se si usa la disposizione automatica delle travature.

## Scrivere la musica in parallelo

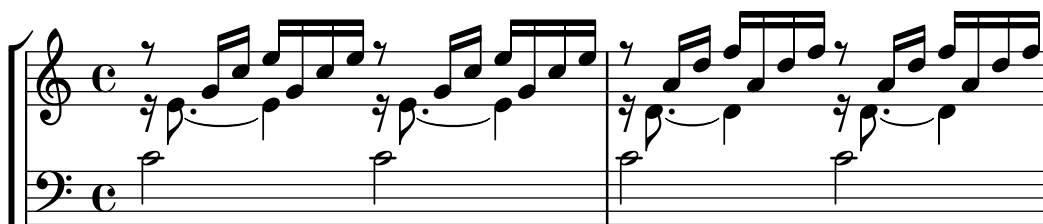
La musica che contiene parti diverse può essere messa in parallelo nel codice di input. La funzione `\parallelMusic` accetta una lista contenente i nomi di un insieme di variabili da creare e un'espressione musicale. Il contenuto delle misure alternate nell'espressione diventa il valore delle rispettive variabili, in modo che possano essere usate successivamente per stampare la musica.

**Nota:** L'uso dei controlli di battuta `|` è obbligatorio e le misure devono avere la stessa durata.

```
\parallelMusic #'(voiceA voiceB voiceC) {
  % Battuta 1
  r8 g'16 c'' e'' g' c'' e'' r8 g'16 c'' e'' g' c'' e'' |
  r16 e'8.~ 4 r16 e'8.~ 4 |
  c'2 c'2 |

  % Battuta 2
  r8 a'16 d'' f'' a' d'' f'' r8 a'16 d'' f'' a' d'' f'' |
  r16 d'8.~ 4 r16 d'8.~ 4 |
  c'2 c'2 |

}
\new StaffGroup <<
  \new Staff << \voiceA \\\ \voiceB >>
  \new Staff { \clef bass \voiceC }
>>
```



L'uso del modo relativo è permesso. Attenzione: il comando `\relative` non deve essere messo dentro `\parallelMusic`. Le note sono relative alla nota precedente della voce, non a quella precedente nell'input. In altre parole, le note relative di `voiceA` ignorano le note in `voiceB`.

```
\parallelMusic #'(voiceA voiceB voiceC) {
  % Battuta 1
  r8 g16 c e g, c e r8 g,16 c e g, c e |
  r16 e8.~ 4 r16 e8.~ 4 |
  c2 c |

  % Battuta 2
  r8 a,16 d f a, d f r8 a,16 d f a, d f |
  r16 d8.~ 4 r16 d8.~ 4 |
  c2 c |
```

```

}
\new StaffGroup <<
  \new Staff << \relative c'' \voiceA \\ \relative c' \voiceB >>
  \new Staff \relative c' { \clef bass \voiceC }
>>

```



Questo è molto utile nella musica per pianoforte. Questo esempio combina sezioni di quattro battute consecutive con quattro variabili:

```

global = {
  \key g \major
  \time 2/4
}

\parallelMusic #'(voiceA voiceB voiceC voiceD) {
  % Battuta 1
  a8    b    c    d    |
  d4          e    |
  c16 d e fis d e fis g |
  a4          a    |

  % Battuta 2
  e8    fis g    a    |
  fis4          g    |
  e16 fis g a fis g a b |
  a4          a    |

  % Bar 3 ...
}

\score {
  \new PianoStaff <<
    \new Staff {
      \global
      <<
        \relative c'' \voiceA
        \\
        \relative c' \voiceB
      >>
    }
    \new Staff {
      \global \clef bass
      <<
        \relative c \voiceC
        \\
        \relative c \voiceD
      >>
    }
  >>
}

```



## Vedi anche

Manuale d'apprendimento: Sezione "Organizzare i brani con le variabili" in *Manuale di Apprendimento*.

Frammenti: Sezione "Simultaneous notes" in *Frammenti di codice*.

## 1.6 Notazione del rigo

Questa sezione spiega come modificare l'aspetto del rigo, come stampare partiture multirigo e come aggiungere indicazioni di tempo e citazioni in corpo più piccolo nel rigo.

### 1.6.1 Aspetto del rigo

Questa sezione presenta i diversi metodi per creare e raggruppare i righi.

## Istanziare nuovi righi

Il *rigo musicale* si crea con i comandi `\new` o `\context`. Ulteriori dettagli in Sezione 5.1.2 [Creating and referencing contexts], pagina 581.

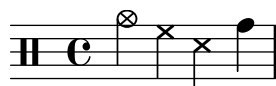
Il contesto di base del rigo è `Staff`:

```
\new Staff \relative { c''4 d e f }
```



Il contesto `DrumStaff` crea un rigo di cinque linee impostato per una tipica batteria. Ogni strumento viene mostrato con un simbolo diverso. Gli strumenti si inseriscono nella modalità percussioni, che si attiva col comando `\drummode`: ogni strumento viene indicato con un nome. Ulteriori dettagli in [Percussion staves], pagina 390.

```
\new DrumStaff {
  \drummode { cymc hh ss tomh }
}
```



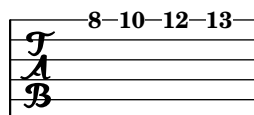
`RhythmicStaff` crea un rigo con una sola linea che mostra soltanto i valori ritmici dell'input. Le durate reali vengono mantenute. Ulteriori dettagli in [Showing melody rhythms], pagina [undefined].

```
\new RhythmicStaff { c4 d e f }
```



`TabStaff` crea un'intavolatura (o tablatura) con sei corde nell'accordatura standard per chitarra. Ulteriori dettagli in [Default tablatures], pagina 341.

```
\new TabStaff \relative { c''4 d e f }
```



Ci sono due contesti del rigo specifici per la notazione di musica antica, `MensuralStaff` e `VaticanaStaff`, descritti in [Pre-defined contexts], pagina 433.

Il contesto `GregorianTranscriptionStaff` crea un rigo per il canto gregoriano moderno. Non mostra le stanghette delle battute.

```
\new GregorianTranscriptionStaff \relative { c''4 d e f e d }
```



Si possono creare nuovi contesti per un singolo rigo, come è spiegato dettagliatamente in Sezione 5.1.6 [Defining new contexts], pagina 594.

## Vedi anche

Glossario musicale: Sezione “rigo” in *Glossario Musicale*,

Guida alla notazione: Sezione 5.1.2 [Creating and referencing contexts], pagina 581, [Percussion staves], pagina 390, <undefined> [Showing melody rhythms], pagina <undefined>, [Default tablatures], pagina 341, [Pre-defined contexts], pagina 433, <undefined> [Staff symbol], pagina <undefined>, [Gregorian chant contexts], pagina 442, [Mensural contexts], pagina 435, Sezione 5.1.6 [Defining new contexts], pagina 594.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

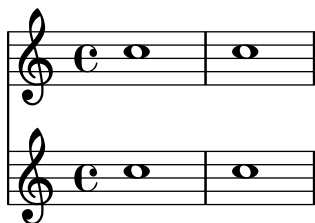
Guida al funzionamento interno: Sezione “Staff” in *Guida al Funzionamento Interno*, Sezione “DrumStaff” in *Guida al Funzionamento Interno*, Sezione “GregorianTranscriptionStaff” in *Guida al Funzionamento Interno*, Sezione “RhythmicStaff” in *Guida al Funzionamento Interno*, Sezione “TabStaff” in *Guida al Funzionamento Interno*, Sezione “MensuralStaff” in *Guida al Funzionamento Interno*, Sezione “VaticanaStaff” in *Guida al Funzionamento Interno*, Sezione “StaffSymbol” in *Guida al Funzionamento Interno*.

## Raggruppare i righi

Esistono vari contesti per raggruppare insieme singoli righi in modo da formare sistemi multirigo. Ogni contesto di raggruppamento imposta il comportamento delle stanghette e lo stile del segno che delimita l’inizio del sistema.

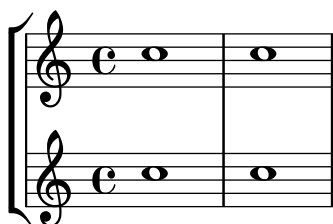
Se non si specifica alcun contesto, vengono usate le proprietà predefinite: il gruppo inizia con una linea verticale e le stanghette non sono collegate.

```
<<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



Nel contesto `StaffGroup`, il gruppo inizia con una parentesi quadra e le stanghette attraversano tutti i righi.

```
\new StaffGroup <<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



Nel contesto `ChoirStaff`, il gruppo inizia con una parentesi quadra, ma le stanghette non sono collegate.

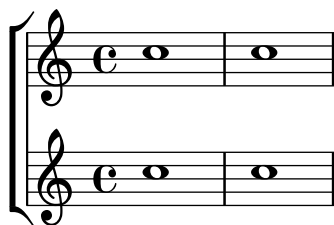
```
\new ChoirStaff <<
```



```

\new Staff \relative { c''1 c }
\new Staff \relative { c''1 c }
>>

```

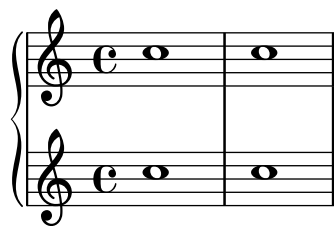


Nel contesto `GrandStaff`, il gruppo inizia con una parentesi graffa e le stanghette sono collegate da rigo a rigo.

```

\new GrandStaff <<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>

```



Il contesto `PianoStaff` è identico a `GrandStaff`, con l'unica differenza che permette di mostrare il nome dello strumento direttamente. Ulteriori dettagli in [\[Instrument names\]](#), pagina [\[undefined\]](#).

```

\new PianoStaff <<
  \set PianoStaff.instrumentName = #"Piano"
  \new Staff \relative { c''1 c }
  \new Staff \relative { \clef bass c1 c }
>>

```



Ogni contesto per il gruppo di righe imposta la proprietà `systemStartDelimiter` su uno dei seguenti valori: `SystemStartBar`, `SystemStartBrace` o `SystemStartBracket`. È presente anche un quarto delimitatore, `SystemStartSquare`, ma deve essere indicato esplicitamente.

Si possono definire nuovi contesti di gruppi di rigo. I dettagli sono spiegati in Sezione 5.1.6 [\[Defining new contexts\]](#), pagina 594.

## Frammenti di codice selezionati

*Usare una parentesi quadra all'inizio di un gruppo di righi*

Si può usare il segno `SystemStartSquare` (uno dei segni che delimitano l'inizio del sistema) impostandolo esplicitamente in un contesto `StaffGroup` o `ChoirStaff`.

```
\score {
  \new StaffGroup { <<
    \set StaffGroup.systemStartDelimiter = #'SystemStartSquare
    \new Staff { c'4 d' e' f' }
    \new Staff { c'4 d' e' f' }
  >> }
}
```



*Mostrare la parentesi anche se c'è un solo rigo nel sistema*

Se c'è un solo rigo in uno dei tipi di rigo `ChoirStaff` o `StaffGroup`, la parentesi e la stanghetta iniziale non appaiono. Si può modificare questo comportamento predefinito sovrascrivendo `collapse-height` e impostando un valore inferiore al numero di linee del rigo.

Nei contesti `PianoStaff` e `GrandStaff`, dove i sistemi iniziano con una parentesi graffa invece di una parentesi quadra, occorre impostare un'altra proprietà, come si vede nel secondo sistema dell'esempio.

```
\score {
  \new StaffGroup <<
    % Must be lower than the actual number of staff lines
    \override StaffGroup.SystemStartBracket.collapse-height = #4
    \override Score.SystemStartBar.collapse-height = #4
    \new Staff {
      c'1
    }
  >>
}

\score {
  \new PianoStaff <<
    \override PianoStaff.SystemStartBrace.collapse-height = #4
    \override Score.SystemStartBar.collapse-height = #4
    \new Staff {
      c'1
    }
  >>
}
```



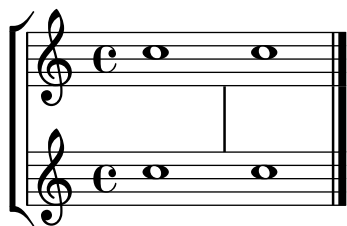


### Formattazione mensurale (stanghette tra i righi)

La formattazione mensurale, in cui le stanghette non appaiono sui righi ma nello spazio tra i righi, si può ottenere usando `StaffGroup` al posto di `ChoirStaff`. La stanghetta sui righi viene nascosta impostando la proprietà `transparent`.

```
global = {
  \hide Staff.BarLine
  s1 s
  % the final bar line is not interrupted
  \undo \hide Staff.BarLine
  \bar "|."
}

\new StaffGroup \relative c' {
  <<
    \new Staff { << \global { c1 c } >> }
    \new Staff { << \global { c c } >> }
  >>
}
```



## Vedi anche

Glossario musicale: Sezione “graffa” in *Glossario Musicale*, Sezione “parentesi quadra” in *Glossario Musicale*, Sezione “accollatura” in *Glossario Musicale*.

Guida alla notazione: `<undefined>` [Instrument names], pagina `<undefined>`, Sezione 5.1.6 [Defining new contexts], pagina 594.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Staff” in *Guida al Funzionamento Interno*, Sezione “StaffGroup” in *Guida al Funzionamento Interno*, Sezione “ChoirStaff” in *Guida al Funzionamento Interno*, Sezione “GrandStaff” in *Guida al Funzionamento Interno*, Sezione “PianoStaff” in *Guida al Funzionamento Interno*, Sezione “SystemStartBar” in *Guida al Funzionamento Interno*, Sezione “SystemStartBrace” in *Guida al Funzionamento Interno*, Sezione “SystemStartBracket” in *Guida al Funzionamento Interno*, Sezione “SystemStartSquare” in *Guida al Funzionamento Interno*.

## Gruppi di righi annidati

I contesti dei gruppi di righi possono essere annidati fino a qualsiasi livello. In questo caso, ogni contesto inferiore crea una nuova parentesi accanto alla parentesi del gruppo superiore.

```
\new StaffGroup <<
  \new Staff \relative { c'2 c | c2 c }
  \new StaffGroup <<
    \new Staff \relative { g'2 g | g2 g }
    \new StaffGroup \with {
```

```

    systemStartDelimiter = #'SystemStartSquare
  }
  <<
    \new Staff \relative { e'2 e | e2 e }
    \new Staff \relative { c'2 c | c2 c }
  >>
>>
>>

```



Si possono definire nuovi gruppi di righe annidati. Ulteriori dettagli in Sezione 5.1.6 [Defining new contexts], pagina 594.

## Frammenti di codice selezionati

### *Annidare i righe*

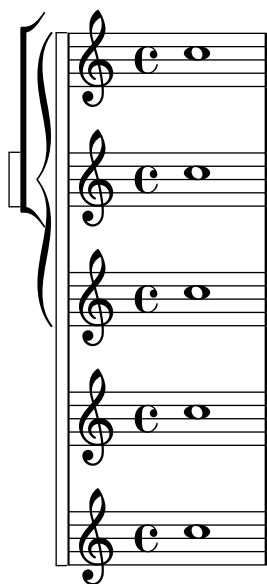
Si può usare la proprietà `systemStartDelimiterHierarchy` per creare gruppi di righe annidati più complessi. Il comando `\set StaffGroup.systemStartDelimiterHierarchy` prende come argomento una lista alfabetica dell'insieme di righe prodotti. Prima di ogni rigo si può assegnare un delimitatore di inizio del sistema. Deve essere racchiuso tra parentesi e collega tutti i righe compresi tra le parentesi. Gli elementi nella lista possono essere omessi, ma la prima parentesi quadra collega sempre tutti i righe. Le possibilità sono `SystemStartBar`, `SystemStartBracket`, `SystemStartBrace` e `SystemStartSquare`.

```

\new StaffGroup
\relative c'' <<
  \set StaffGroup.systemStartDelimiterHierarchy
    = #'(SystemStartSquare (SystemStartBrace (SystemStartBracket a
                                          (SystemStartSquare b) ) c ) d)

  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
>>

```



## Vedi anche

Guida alla notazione: [\[Grouping staves\]](#), pagina [\[Instrument names\]](#), pagina [\[Defining new contexts\]](#), sezione 5.1.6, pagina 594.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “StaffGroup” in *Guida al Funzionamento Interno*, Sezione “ChoirStaff” in *Guida al Funzionamento Interno*, Sezione “SystemStartBar” in *Guida al Funzionamento Interno*, Sezione “SystemStartBrace” in *Guida al Funzionamento Interno*, Sezione “SystemStartBracket” in *Guida al Funzionamento Interno*, Sezione “SystemStartSquare” in *Guida al Funzionamento Interno*.

## Separare i sistemi

Se il numero di sistemi per pagina cambia di pagina in pagina, è consuetudine separare i sistemi con un segno separatore. Per impostazione predefinita questo segno è disattivo, ma può essere attivato con un’opzione in `\paper`.

```
\book {
  \score {
    \new StaffGroup <<
      \new Staff {
        \relative {
          c''4 c c c
          \break
          c4 c c c
        }
      }
      \new Staff {
        \relative {
          c''4 c c c
          \break
          c4 c c c
        }
      }
    >>
  }
  \paper {
    system-separator-markup = \slashSeparator
  }
}
```

```

% i seguenti comandi servono soltanto alla formattazione di questa documentazione
paper-width = 100\mm
paper-height = 100\mm
tagline = ##f
}
}

```



## Vedi anche

Guida alla notazione: [\[Page layout\]](#), pagina [\[Page layout\]](#).

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

### 1.6.2 Modificare singoli righi

Questa sezione spiega come modificare gli attributi specifici di un rigo, per esempio il numero di linee o la dimensione del rigo. Vengono descritti anche i metodi per iniziare e finire un rigo e per impostare le sezioni ossia.

#### Simbolo del rigo

I comandi `\stopStaff` e `\startStaff` servono a fermare o (ri)avviare le linee del rigo, per impedire che appaiano in un punto della partitura.

```

\relative {
  \stopStaff f'4 d \startStaff g, e
  f'4 d \stopStaff g, e
  f'4 d \startStaff g, e
}

```



#### Comandi predefiniti

`\startStaff`, `\stopStaff`.

Le linee di un rigo appartengono all'oggetto `StaffSymbol` (che comprende i tagli aggiuntivi) e si possono modificare tramite le proprietà di `StaffSymbol`; però queste modifiche devono essere fatte prima che il rigo sia (ri)avviato.

Si può cambiare il numero di linee del rigo:

```
\relative {
  f''4 d \stopStaff
  \override Staff.StaffSymbol.line-count = #2
  \startStaff g, e |

  f'4 d \stopStaff
  \revert Staff.StaffSymbol.line-count
  \startStaff g, e |
}
```



Si può cambiare anche la posizione di ogni linea del rigo. Un elenco di numeri definisce la posizione di ogni linea. I valori consueti sono 0 per la linea centrale e (-4 -2 0 2 4) per le altre. La linea del rigo appare solo se è presente il suo valore, quindi questo comando permette di variare anche il numero delle linee, oltre alla loro posizione.

```
\relative {
  f''4 d \stopStaff
  \override Staff.StaffSymbol.line-positions = #'(1 3 5 -1 -3)
  \startStaff g, e |
  f'4 d \stopStaff
  \override Staff.StaffSymbol.line-positions = #'(8 6.5 -6 -8 -0.5)
  \startStaff g, e |
}
```



Per conservare le tipiche direzioni dei gambi (nella metà inferiore del rigo i gambi puntano in su, mentre in quella superiore sono rivolti in giù), occorre allineare la linea centrale (o lo spazio) del rigo personalizzato alla posizione della linea centrale normale (0). Potrà essere necessario regolare la posizione della chiave e del Do centrale per adattarsi alle nuove linee. Si veda `\undefined` [Clef], pagina `\undefined`.

Si può modificare lo spessore della linea del rigo. Per impostazione predefinita, questa modifica ha effetto anche sui tagli addizionali e sui gambi.

```
\new Staff \with {
  \override StaffSymbol.thickness = #3
} \relative {
  f''4 d g, e
}
```



È anche possibile impostare lo spessore dei tagli addizionali in modo indipendente dalle linee del rigo.

```
\new Staff \with {
```

```

\override StaffSymbol.thickness = #2
\override StaffSymbol.ledger-line-thickness = #'(0.5 . 0.4)
} \relative {
  f'''4 a, a,, f
}

```



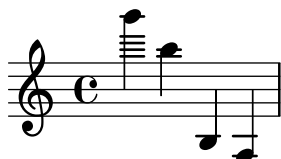
Il primo valore viene moltiplicato per lo spessore della linea del rigo, il secondo per la spaziatura del rigo; la somma dei due valori definisce il nuovo valore dello spessore del taglio addizionale.

Si possono modificare le posizioni verticali dei tagli addizionali:

```

\new Staff \with {
  \override StaffSymbol.ledger-positions = #'(-3 -2 -1 2 5 6)
} \relative {
  f'''4 a, a,, f
}

```



Si possono far apparire ulteriori tagli addizionali sopra o sotto le teste delle note, a seconda della posizione corrente relativa alle altre teste, anch'esse con i propri tagli addizionali.

```

\new Staff \with {
  \override StaffSymbol.ledger-extra = #4
} \relative {
  f'''4 a, d, f,
}

```



Si possono far apparire i tagli addizionali anche dentro il rigo quando servono delle linee personalizzate. L'esempio mostra la posizione predefinita dei tagli addizionali quando la proprietà `ledger-position` è impostata e quando non lo è. Nell'esempio il comando `\stopStaff` serve ad annullare il comando `\override` per l'oggetto `StaffSymbol`.

```

\relative d' {
  \override Staff.StaffSymbol.line-positions = #'(-8 0 2 4)
  d4 e f g
  \stopStaff
  \startStaff
  \override Staff.StaffSymbol.ledger-positions = #'(-8 -6 (-4 -2) 0)
  d4 e f g
}

```





Si può cambiare la distanza tra le linee del rigo. Tale modifica ha effetto anche sulla spaziatura della linea.

```
\new Staff \with {
  \override StaffSymbol.staff-space = #1.5
} \relative {
  f''4 d, g, e,
}
```



## Frammenti di codice selezionati

*Rendere alcune linee del rigo più spesse delle altre*

In ambito didattico può essere utile rendere più spesso una linea del rigo (per esempio, la linea centrale, o per sottolineare la linea della chiave di Sol). Per farlo si possono aggiungere altre linee e posizionarle molto vicino alla linea che deve essere evidenziata, usando la proprietà `line-positions` dell'oggetto `StaffSymbol`.

```
{
  \override Staff.StaffSymbol.line-positions =
    #'(-4 -2 -0.2 0 0.2 2 4)
  d'4 e' f' g'
}
```



## Vedi anche

Glossario musicale: Sezione “linea” in *Glossario Musicale*, Sezione “taglio addizionale” in *Glossario Musicale*, Sezione “rigo (o pentagramma)” in *Glossario Musicale*.

Guida alla notazione: `<undefined>` [Clef], pagina `<undefined>`.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “StaffSymbol” in *Guida al Funzionamento Interno*, Sezione “staff-symbol-interface” in *Guida al Funzionamento Interno*.

## Righi ossia

I righi ossia si possono creare aggiungendo un nuovo rigo simultaneo nel punto giusto:

```
\new Staff \relative {
  c'4 b d c
  <<
  { c4 b d c }
  \new Staff { e4 d f e }
```

```
>>
c4 b c2
}
```



Tuttavia, questo esempio non produce quel che normalmente si desidera. Per creare righi ossia che siano sopra il rigo originale, non abbiano indicazione di tempo né chiave e abbiano un tipo di carattere più piccolo, sono necessarie delle modifiche manuali. Il Manuale d'apprendimento descrive una tecnica specifica per ottenere questo risultato, a partire da Sezione “Annidare le espressioni musicali” in *Manuale di Apprendimento*.

L'esempio seguente usa la proprietà `alignAboveContext` per allineare il rigo ossia. Questo metodo conviene quando sono necessari solo pochi righi ossia.

```
\new Staff = "main" \relative {
  c' '4 b d c
  <<
    { c4 b d c }

    \new Staff \with {
      \remove "Time_signature_engraver"
      alignAboveContext = #"main"
      \magnifyStaff #2/3
      firstClef = ##f
    }
    { e4 d f e }
  >>
  c4 b c2
}
```



Se si hanno molti righi ossia isolati, è meglio creare un contesto `Staff` vuoto con un *identificativo del contesto* specifico; i righi ossia possono essere creati *chiamando* questo contesto e usando `\startStaff` e `\stopStaff` nei punti richiesti. I vantaggi di questo metodo sono più evidenti se il brano è più lungo del seguente esempio.

```
<<
\new Staff = "ossia" \with {
  \remove "Time_signature_engraver"
  \hide Clef
  \magnifyStaff #2/3
}
{ \stopStaff s1*6 }
```

```

\new Staff \relative {
  c'4 b c2
  <<
    { e4 f e2 }
    \context Staff = "ossia" {
      \startStaff e4 g8 f e2 \stopStaff
    }
  >>
  g4 a g2 \break
  c4 b c2
  <<
    { g4 a g2 }
    \context Staff = "ossia" {
      \startStaff g4 e8 f g2 \stopStaff
    }
  >>
  e4 d c2
}
>>

```



4



Come alternativa, si può usare il comando `\Staff \RemoveEmptyStaves` per creare i righi ossia. Questo metodo conviene quando i righi ossia si trovano subito dopo un'interruzione di linea. Ulteriori informazioni su `\Staff \RemoveEmptyStaves` si trovano in [\[Hiding staves\]](#), pagina [\[Hiding staves\]](#).

```

<<
  \new Staff = "ossia" \with {
    \remove "Time_signature_engraver"
    \hide Clef
    \magnifyStaff #2/3
  } \relative {
    R1*3
    c'4 e8 d c2
  }
  \new Staff \relative {
    c'4 b c2
    e4 f e2
    g4 a g2 \break
  }

```

```

c4 b c2
g4 a g2
e4 d c2
}
>>

\layout {
  \context {
    \Staff \RemoveEmptyStaves
    \override VerticalAxisGroup.remove-first = ##t
  }
}

```



## Frammenti di codice selezionati

*Allineare verticalmente gli ossia e il testo vocale*

Questo frammento mostra come usare le proprietà di contesto `alignBelowContext` e `alignAboveContext` per controllare il posizionamento del testo vocale e degli ossia.

```

\paper {
  ragged-right = ##t
}

\relative c' <<
  \new Staff = "1" { c4 c s2 }
  \new Staff = "2" { c4 c s2 }
  \new Staff = "3" { c4 c s2 }
  { \skip 2
    <<
      \lyrics {
        \set alignBelowContext = #"1"
        lyrics4 below
      }
      \new Staff \with {
        alignAboveContext = #"3"
        fontSize = #-2
        \override StaffSymbol.staff-space = #(magstep -2)
        \remove "Time_signature_engraver"
      } {
        \tuplet 6/4 {
          \override TextScript.padding = #3

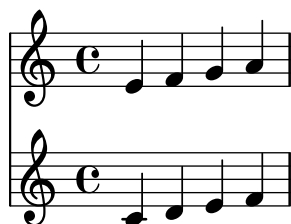
```



**Nota:** Un rigo viene considerato vuoto quando contiene soltanto pause multiple, pause, salti, pause spaziatrici o una combinazione di questi elementi.

```
\layout {
  \context {
    \Staff \RemoveEmptyStaves
  }
}

\relative <<
  \new Staff {
    e'4 f g a \break
    b1 \break
    a4 b c2
  }
  \new Staff {
    c,4 d e f \break
    R1 \break
    f4 g c,2
  }
}>>
```



`\Staff \RemoveEmptyStaves` si può usare anche per creare sezioni ossia per un rigo. I dettagli si trovano in [\[Ossia staves\]](#), pagina [\[undefined\]](#).

Per nascondere i righi vuoti nei contesti della musica antica si può usare il comando `\VaticanaStaff \RemoveEmptyStaves`. Analogamente, `\RhythmicStaff \RemoveEmptyStaves` permette di nascondere i contesti `RhythmicStaff` vuoti.

## Comandi predefiniti

`\Staff \RemoveEmptyStaves`, `\VaticanaStaff \RemoveEmptyStaves`, `\RhythmicStaff \RemoveEmptyStaves`.

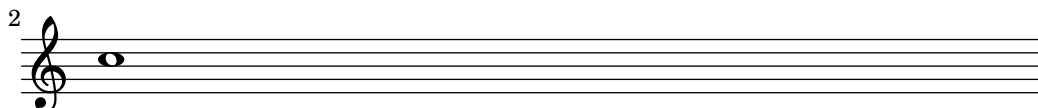
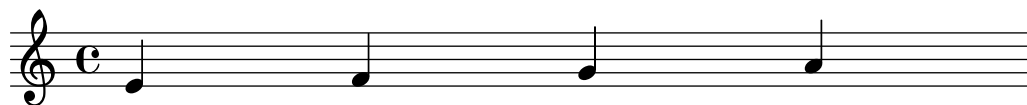
## Frammenti di codice selezionati

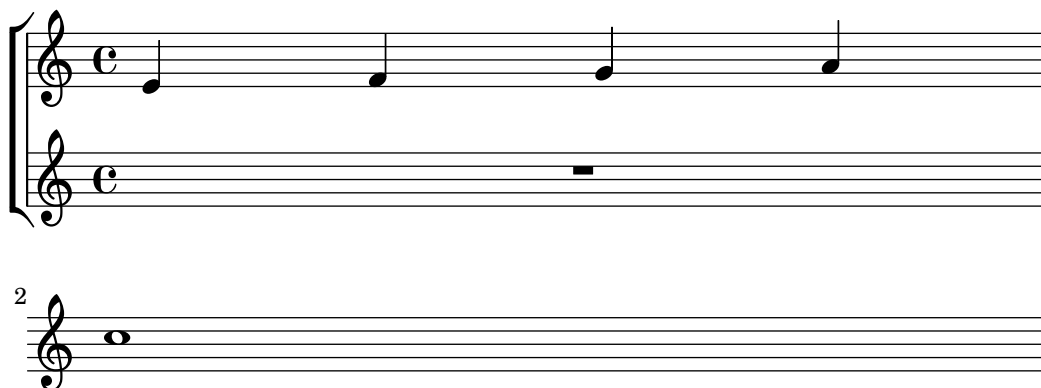
*Eliminare la prima linea vuota*

Il primo rigo vuoto si può togliere dalla partitura impostando la proprietà `remove-first` di `VerticalAxisGroup`. Questa impostazione agisce a livello globale se posta nel blocco `\layout`, a livello locale se posta nel rigo specifico che deve essere tolto. Nel secondo caso, si deve specificare il contesto (`Staff` si applica solo al rigo corrente) prima della proprietà.

Il rigo inferiore del secondo gruppo di righe non viene rimosso, perché l'impostazione ha effetto solo sul rigo in cui si trova.

```
\layout {
  \context {
    \Staff \RemoveEmptyStaves
    % To use the setting globally, uncomment the following line:
    % \override VerticalAxisGroup.remove-first = ##t
  }
}
\new StaffGroup <<
  \new Staff \relative c' {
    e4 f g a \break
    c1
  }
  \new Staff {
    % To use the setting globally, comment this line,
    % uncomment the line in the \layout block above
    \override Staff.VerticalAxisGroup.remove-first = ##t
    R1 \break
    R
  }
}
>>
\new StaffGroup <<
  \new Staff \relative c' {
    e4 f g a \break
    c1
  }
  \new Staff {
    R1 \break
    R
  }
}
>>
```





## Vedi anche

Glossario musicale: Sezione “rigo temporaneo” in *Glossario Musicale*.

Manuale d’apprendimento: Sezione “Visibilità e colore degli oggetti” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.1.5 [Changing context default settings], pagina 589, <undefined> [Staff symbol], pagina <undefined>, <undefined> [Ossia staves], pagina <undefined>, <undefined> [Hidden notes], pagina <undefined>, <undefined> [Invisible rests], pagina <undefined>, Sezione 5.4.7 [Visibility of objects], pagina 618.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “ChordNames” in *Guida al Funzionamento Interno*, Sezione “FiguredBass” in *Guida al Funzionamento Interno*, Sezione “Lyrics” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*, Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*, Sezione “Staff\_symbol\_engraver” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Se si toglie l’incisore `Staff_symbol_engraver` vengono nascoste anche le stanghette. Se si forza la visibilità delle stanghette, potrebbero verificarsi degli errori di formattazione. In questo caso, conviene usare i seguenti comandi invece di togliere l’incisore:

```
\omit StaffSymbol
\override NoteHead.no-ledgers = ##t
```

Per i problemi noti e gli avvertimenti relativi a `\Staff \RemoveEmptyStaves` si veda Sezione 5.1.5 [Changing context default settings], pagina 589.

### 1.6.3 Scrittura delle parti

Questa sezione spiega come inserire in una partitura le indicazioni di tempo e i nomi degli strumenti. Mostra anche come citare altre voci e come formattare le citazioni in corpo più piccolo.

## Nomi degli strumenti

I nomi degli strumenti possono essere fatti apparire, alla sinistra dei righi, nei contesti `Staff`, `PianoStaff`, `StaffGroup`, `GrandStaff` e `ChoirStaff`. Il valore di `instrumentName` viene usato per il primo rigo e quello di `shortInstrumentName` per tutti i righi successivi.

```
\new Staff \with {
  instrumentName = #"Violin "
  shortInstrumentName = #"Vln. "
} \relative {
  c'4.. g'16 c4.. g'16 \break | c1 |
}
```





Si può usare `\markup` per creare nomi più complessi:

```
\new Staff \with {
  instrumentName = \markup {
    \column { "Clarineti"
      \line { "in B" \smaller \flat }
    }
  }
} \relative {
  c''4 c,16 d e f g2
}
```



Se due o più contesti del rigo sono raggruppati insieme, i nomi degli strumenti, sia quello normale che quello abbreviato, vengono centrati automaticamente. Per allineare al centro i nomi degli strumenti che vanno a capo, occorre usare `\center-column`:

```
<<
  \new Staff \with {
    instrumentName = #"Flute"
  } \relative {
    f''2 g4 f
  }
  \new Staff \with {
    instrumentName = \markup {
      \center-column { "Clarinet"
        \line { "in B" \smaller \flat }
      }
    }
  } \relative { c''4 b c2 }
>>
```



Tuttavia, se i nomi degli strumenti sono lunghi, potranno essere centrati solo aumentando i valori di `indent` e `short-indent`. Ulteriori dettagli su queste impostazioni si trovano in `\paper variables for shifts and indents`, pagina `\paper variables for shifts and indents`.

```
<<
  \new Staff \with {
```

```

    instrumentName = #"Alto Flute in G"
    shortInstrumentName = #"Flt."
} \relative {
    f''2 g4 f \break
    g4 f g2
}
\new Staff \with {
    instrumentName = #"Clarinet"
    shortInstrumentName = #"Clar."
} \relative {
    c''4 b c2 \break
    c2 b4 c
}
>>

\layout {
    indent = 3.0\cm
    short-indent = 1.5\cm
}

```

Alto Flute in G

Clarinet

Flt.

Clar.

Per impostare i nomi degli strumenti in altri contesti (come `ChordNames` o `FiguredBass`), si deve aggiungere l'incisore `Instrument_name_engraver` a quel contesto. Ulteriori dettagli in Sezione 5.1.4 [Modifying context plug-ins], pagina 587.

`shortInstrumentName` può essere cambiato all'interno di un brano, insieme a altre impostazioni necessarie al nuovo strumento. Tuttavia, di `instrumentName` apparirà solo la prima definizione e le modifiche successive saranno ignorate:

```

prepPiccolo = <>^\markup \italic { muta in Piccolo }

setPiccolo = {
    \set Staff.instrumentName = #"Piccolo"
    \set Staff.shortInstrumentName = #"Picc."
    \set Staff.midiInstrument = #"piccolo"
    <>^\markup \bold { Piccolo }
    \transposition c''
}

```

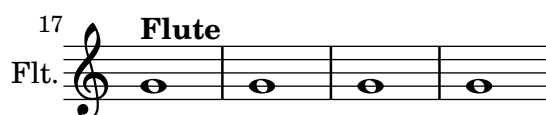
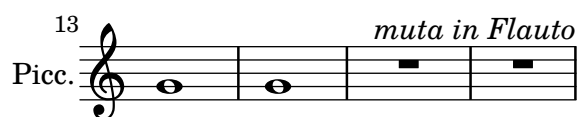
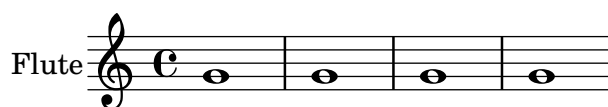
```

prepFlute = <>^\markup \italic { muta in Flauto }

setFlute = {
  \set Staff.instrumentName = #"Flute"
  \set Staff.shortInstrumentName = #"Flt."
  \set Staff.midiInstrument = #"flute"
  <>^\markup \bold { Flute }
  \transposition c'
}

\new Staff \with {
  instrumentName = #"Flute"
  shortInstrumentName = #"Flt."
  midiInstrument = #"flute"
}
\relative {
  g'1 g g g \break
  g1 g \prepPiccolo R R \break
  \setPiccolo
  g1 g g g \break
  g1 g \prepFlute R R \break
  \setFlute
  g1 g g g
}

```



## Vedi anche

Guida alla notazione: `\paper variables for shifts and indents`, pagina `\undefined`, Sezione 5.1.4 `[Modifying context plug-ins]`, pagina 587.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “InstrumentName” in *Guida al Funzionamento Interno*, Sezione “PianoStaff” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*.

## Citare altre voci

È molto comune che una voce usi le stesse note di un'altra voce. Per esempio, il primo e il secondo violino che suonano la stessa frase durante un particolare passaggio del brano. Per evitare di reinserire la musica di nuovo per la seconda voce, si può far sì che una voce *citi* l'altra.

Il comando `\addQuote`, usato nell'ambito di più alto livello, definisce un flusso musicale da cui poter citare i frammenti.

Il comando `\quoteDuring` serve a indicare il punto in cui inizia la citazione. È seguito da due argomenti: il nome della voce citata, come è definito da `\addQuote`, e un'espressione musicale per la durata della citazione.

```
fluteNotes = \relative {
  a'4 gis g gis | b4~"quoted" r8 ais\p a4( f)
}

oboeNotes = \relative {
  c''4 cis c b \quoteDuring #"flute" { s1 }
}

\addQuote "flute" { \fluteNotes }

\score {
  <<
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}
```



Se l'espressione musicale usata in `\quoteDuring` contiene note invece di pause spaziatrici o multiple, la citazione apparirà in forma polifonica e potrebbe causare risultati indesiderati.

```
fluteNotes = \relative {
  a'4 gis g gis | b4~"quoted" r8 ais\p a4( f)
}

oboeNotes = \relative {
  c''4 cis c b \quoteDuring #"flute" { e4 r8 ais b4 a }
}
```

```

}

\addQuote "flute" { \fluteNotes }

\score {
  <<
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}

```



Se un comando `\unfoldRepeat` in un'espressione musicale deve essere stampato quando si usa `\quoteDuring`, allora anch'esso deve contenere il suo comando `\unfoldRepeat`;

```

fluteNotes = \relative {
  \repeat volta 2 { a'4 gis g gis }
}

oboeNotesDW = \relative {
  \repeat volta 2 \quoteDuring #"incorrect" { s1 }
}

oboeNotesW = \relative {
  \repeat volta 2 \quoteDuring #"correct" { s1 }
}

\addQuote "incorrect" { \fluteNotes }

\addQuote "correct" { \unfoldRepeats \fluteNotes }

\score {
  \unfoldRepeats
  <<
    \new Staff \with { instrumentName = "Flute" }
    \fluteNotes
    \new Staff \with { instrumentName = "Oboe (incorrect)" }
    \oboeNotesDW
    \new Staff \with { instrumentName = "Oboe (correct)" }
    \oboeNotesW
  >>
}

```

Flute

Oboe (incorrect)

Oboe (correct)

Il comando `\quoteDuring` usa le impostazioni `\transposition` sia della parte citata sia di quella che cita per produrre delle note per la parte che cita che abbiano la stessa altezza di quelle nella parte citata.

```
clarinetNotes = \relative c'' {
  \transposition bes
  \key d \major
  b4 ais a ais | cis4^"quoted" r8 bis\p b4( f)
}

oboeNotes = \relative {
  c''4 cis c b \quoteDuring #"clarinet" { s1 }
}

\addQuote "clarinet" { \clarinetNotes }

\score {
  <<
    \new Staff \with { instrumentName = "Clarinet" } \clarinetNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}
```

Clarinet

Oboe

La musica citata include tutte le articolazioni, dinamiche, annotazioni, etc. presenti nel frammento citato. È possibile scegliere quali di questi oggetti far apparire usando la proprietà di contesto `quotedEventTypes`.

```
fluteNotes = \relative {
  a'2 g2 |
  b4\<^"quoted" r8 ais a4\f( c->)
}

oboeNotes = \relative {
  c''2. b4 |
  \quoteDuring #"flute" { s1 }
}
```

```

\addQuote "flute" { \fluteNotes }

\score {
  <<
    \set Score.quotedEventTypes = #'(note-event articulation-event
                                   crescendo-event rest-event
                                   slur-event dynamic-event)
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}

```



Le citazioni possono anche essere contrassegnate; si veda [\[Using tags\]](#), pagina [\[undefined\]](#).

## Vedi anche

Guida alla notazione: [\[Instrument transpositions\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Using tags\]](#), pagina [\[undefined\]](#).

File installati: `scm/define-event-classes.scm`.

Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Music classes” in *Guida al Funzionamento Interno*, Sezione “QuoteMusic” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Solo il contenuto della prima voce in un comando `\addQuote` sarà preso in considerazione per la citazione; quindi se l'espressione musicale contiene comandi `\new` o `\context Voice`, il loro contenuto non verrà citato. La citazione degli abbellimenti non è supportata e potrebbe causare il crash di LilyPond; la citazione di terzine annidate potrebbe produrre una notazione mediocre.

## Formattazione delle notine

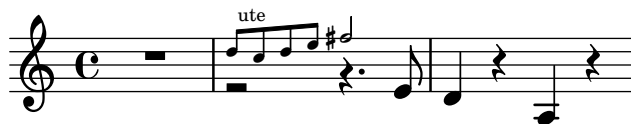
Il modo più semplice per formattare le notine è creare esplicitamente un contesto `CueVoice` all'interno della parte.

```

\relative {
  R1
  <<
    { e'2\rest r4. e8 }
    \new CueVoice {
      \stemUp d'8~"flute" c d e fis2
    }
  >>
}

```

```
d,4 r a r
}
```



Si può usare il comando `\cueClef` all'interno di un contesto `CueVoice` esplicito se è richiesto un cambiamento di chiave; in questo modo la chiave apparirà nella dimensione giusta per le notine. Si può poi usare il comando `\cueClefUnset` per tornare alla chiave originale, di nuovo nella dimensione giusta.

```
\relative {
  \clef "bass"
  R1
  <<
    { e'2\rest r4. \cueClefUnset e,8 }
    \new CueVoice {
      \cueClef "treble" \stemUp d''8^"flute" c d e fis2
    }
  >>
  d,,4 r a r
}
```



I comandi `\cueClef` e `\cueClefUnset` si possono usare anche senza un esplicito contesto `CueVoice`.

```
\relative {
  \clef "bass"
  R1
  \cueClef "treble"
  d''8^"flute" c d e fis2
  \cueClefUnset
  d,,4 r a r
}
```



Per posizionamenti complessi delle notine, per esempio includere la trasposizione o inserire delle notine da varie sorgenti musicali, si possono usare i comandi `\cueDuring` o `\cueDuringWithClef`. Questi sono delle varianti più specializzate di `\quoteDuring`, introdotto in [\[Quoting other voices\]](#), pagina [\[undefined\]](#), nella sezione precedente.

La sintassi è:

```
\cueDuring #nomecitazione #direzione #musica
```

e

```
\cueDuringWithClef #nomecitazione #direzione #chiave #musica
```



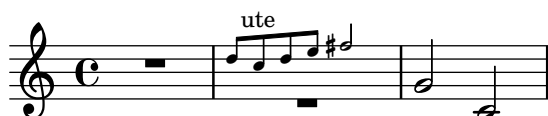
La musica delle misure che corrispondono a *nomecitazione* viene aggiunta in un contesto *CueVoice* e si colloca in simultanea con *musica*, creando quindi una situazione polifonica. La *direzione* prende l'argomento UP o DOWN, e corrisponde alla prima e alla seconda voce rispettivamente, determinando come le notine appaiono in relazione all'altra voce.

```
fluteNotes = \relative {
  r2. c''4 | d8 c d e fis2 | g2 d |
}

oboeNotes = \relative c'' {
  R1
  <>^\markup \tiny { flute }
  \cueDuring #"flute" #UP { R1 }
  g2 c,
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \oboeNotes
}
```



È possibile controllare quali aspetti della musica vengono citati con `\cueDuring` impostando la proprietà `quotedCueEventTypes`. Il suo valore predefinito è `'(note-event rest-event tie-event beam-event tuplet-span-event)`, che significa che vengono citati solo note, pause, legature di valore, travature e gruppi irregolari, ma non le articolazioni, le indicazioni dinamiche, il testo a margine, etc.

**Nota:** Quando una voce inizia con `\cueDuring`, come nell'esempio seguente, il contesto *Voice* deve essere dichiarato esplicitamente, altrimenti l'intera espressione musicale appartiene al contesto *CueVoice*.

```
oboeNotes = \relative {
  r2 r8 d''16(\f f e g f a)
  g8 g16 g g2.
}

\addQuote "oboe" { \oboeNotes }

\new Voice \relative c'' {
  \set Score.quotedCueEventTypes = #'(note-event rest-event tie-event
                                     beam-event tuplet-span-event
                                     dynamic-event slur-event)

  \cueDuring #"oboe" #UP { R1 }
  g2 c,
}
```



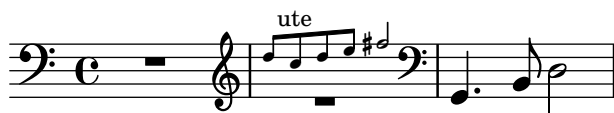
Si può usare il comando `\markup` per mostrare il nome dello strumento citato. Se le citazioni in corpo più piccolo richiedono un cambio di chiave, si può fare manualmente, ma anche il ripristino della chiave originale dovrà essere fatto manualmente al termine delle citazioni.

```
fluteNotes = \relative {
  r2. c''4 d8 c d e fis2 g2 d2
}

bassoonNotes = \relative c {
  \clef bass
  R1
  \clef treble
  <>^\markup \tiny { flute }
  \cueDuring #"flute" #UP { R1 }
  \clef bass
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}
```



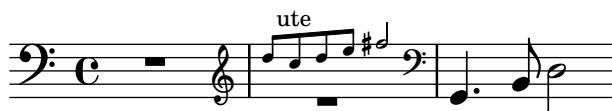
Altrimenti si può usare la funzione `\cueDuringWithClef`. Questo comando prende un ulteriore argomento per specificare il cambio di chiave da usare per le citazioni in corpo più piccolo ma mostrerà automaticamente la chiave originale appena le citazioni sono finite.

```
fluteNotes = \relative {
  r2. c''4 d8 c d e fis2 g2 d2
}

bassoonNotes = \relative c {
  \clef bass
  R1
  <>^\markup { \tiny "flute" }
  \cueDuringWithClef #"flute" #UP #"treble" { R1 }
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}
```



Come `\quoteDuring`, `\cueDuring` prende in considerazione la trasposizione degli strumenti. Le citazioni in corpo più piccolo vengono mostrate nelle altezze necessarie allo strumento che riprende la citazione per riprodurre gli stessi suoni dello strumento citato.

Per trasporre le citazioni in corpo più piccolo in modo diverso, si usa `\transposedCueDuring`. Questo comando prende un ulteriore argomento per specificare (in modalità assoluta) l'altezza da usare nella partitura per rappresentare il Do centrale in intonazione reale. È utile nel caso di citazioni da uno strumento che ha un registro completamente diverso.

```
piccoloNotes = \relative {
  \clef "treble^8"
  R1
  c''^8 c c e g2
  c4 g g2
}

bassClarinetNotes = \relative c' {
  \key d \major
  \transposition bes,
  d4 r a r
  \transposedCueDuring #"piccolo" #UP d { R1 }
  d4 r a r
}

\addQuote "piccolo" { \piccoloNotes }

<<
  \new Staff \piccoloNotes
  \new Staff \bassClarinetNotes
>>
```



Il comando `\killCues` toglie le citazioni in corpo più piccolo da un'espressione musicale, in modo che la stessa espressione musicale possa essere usata per produrre sia la parte strumentale con le citazioni in corpo più piccolo sia l'intera partitura. Il comando `\killCues` toglie soltanto le note e gli eventi citati da `\cueDuring`. Altre annotazioni relative alle citazioni in corpo più piccolo, come i cambi di chiave e il nome che identifica lo strumento sorgente, possono essere contrassegnate per includerle in modo selettivo nella partitura; si veda `\killCues` [Using tags], pagina `\killCues`.

```
fluteNotes = \relative {
  r2. c''^4 d8 c d e fis2 g2 d2
}

bassoonNotes = \relative c {
  \clef bass
```

```

R1
\tag #'part {
  \clef treble
  <>^\markup { \tiny "flute" }
}
\cueDuring #"flute" #UP { R1 }
\tag #'part \clef bass
g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}

\new StaffGroup <<
  \new Staff {
    \fluteNotes
  }
  \new Staff {
    \removeWithTag #'part { \killCues { \bassoonNotes } }
  }
>>

```



Altrimenti, i cambi di chiave e i nomi identificativi degli strumenti possono essere inseriti in una definizione, in modo da poterli riutilizzare, col comando `\addInstrumentDefinition` descritto in [\(undefined\)](#) [Instrument names], pagina [\(undefined\)](#).

## Vedi anche

Guida alla notazione: [\(undefined\)](#) [Quoting other voices], pagina [\(undefined\)](#), [\(undefined\)](#) [Instrument transpositions], pagina [\(undefined\)](#), [\(undefined\)](#) [Instrument names], pagina [\(undefined\)](#), [\(undefined\)](#) [Clef], pagina [\(undefined\)](#), [\(undefined\)](#) [Musical cues], pagina [\(undefined\)](#), [\(undefined\)](#) [Using tags], pagina [\(undefined\)](#).

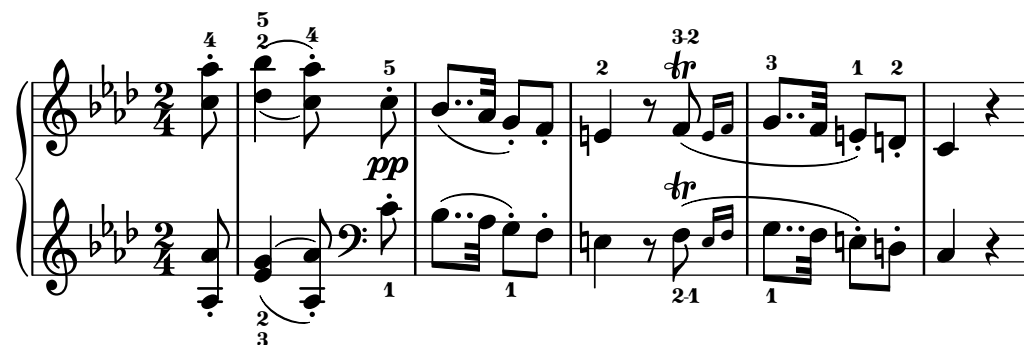
Frammenti: Sezione “Staff notation” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “CueVoice” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Quando si usa `\cueDuring`, si possono verificare delle collisioni tra le pause nel contesto `Voice` e quelle in `CueVoice`. Quando si usa `\cueDuringWithClef` o `\transposedCueDuring`, l'argomento ulteriore richiesto da entrambi deve venire dopo la citazione e la direzione.

### 1.7 Note editoriali



Questa sezione tratta dei vari modi con cui cambiare l'aspetto delle note e aggiungere un'analisi o un accento didattico.

#### 1.7.1 Interne al rigo

Questa sezione spiega come aggiungere enfasi agli elementi interni al rigo.

### Scelta della dimensione del tipo di carattere

#### Nota:

Per le dimensioni del testo, leggere [\[Selecting font and font size\]](#), pagina [\[undefined\]](#).  
 Per la dimensione del rigo, leggere [\[Setting the staff size\]](#), pagina [\[undefined\]](#).  
 Per le citazioni in corpo piccolo, leggere [\[Formatting cue notes\]](#), pagina [\[undefined\]](#).  
 Per i righi ossia, leggere [\[Ossia staves\]](#), pagina [\[undefined\]](#).

Per modificare la dimensione di un elemento della notazione senza cambiare anche la dimensione del rigo, si può specificare un fattore di ingrandimento col comando `\magnifyMusic`:

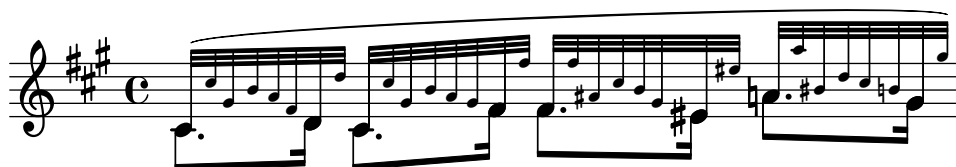
```
\new Staff <<
  \new Voice \relative {
    \voiceOne
    <e' e'>4 <f f'>8. <g g'>16 <f f'>8 <e e'>4 r8
  }
  \new Voice \relative {
    \voiceTwo
    \magnifyMusic 0.63 {
      \override Score.SpacingSpanner.spacing-increment = #(* 1.2 0.63)
      r32 c'' a c a c a c r c a c a c a c
      r c a c a c a c a c a c a c a c
    }
  }
>>
```



L'\code{override} in questo esempio serve a eludere un difetto del programma, spiegato in “Known issues and warnings” alla fine di questa sezione.

Se la testa di una nota di dimensione normale è accorpata con una più piccola, potrebbe essere necessario ripristinare la dimensione della nota più piccola (con '\code{once \normalsize}') in modo che i gambi e le alterazioni siano allineati correttamente:

```
\new Staff <<
  \key fis \minor
  \mergeDifferentlyDottedOn
  \new Voice \relative {
    \voiceOne
    \magnifyMusic 0.63 {
      \override Score.SpacingSpanner.spacing-increment = #(* 1.2 0.63)
      \once \normalsize cis'32( cis' gis b a fis \once \normalsize d d'
      \once \normalsize cis, cis' gis b a gis \once \normalsize fis fis'
      \once \normalsize fis, fis' ais, cis b gis \once \normalsize eis eis'
      \once \normalsize a, a' bis, d cis b \once \normalsize gis gis')
    }
  }
  \new Voice \relative {
    \voiceTwo
    cis'8. d16 cis8. fis16 fis8. eis16 a8. gis16
  }
>>
```



Il comando \code{magnifyMusic} non è adatto per le citazioni in corpo piccolo, gli abbellimenti o i righi ossia, per i quali esistono metodi di inserimento più appropriati. È invece utile quando la dimensione della notazione cambia in una singola parte strumentale su un rigo e quando gli abbellimenti non sono la scelta appropriata, come nei passaggi di tipo cadenza o in casi simili agli esempi precedenti. Impostando il valore di \code{magnifyMusic} su 0.63 si duplicano le dimensioni del contesto CueVoice.

**Nota:** Il comando \code{magnifyMusic} *non* deve essere usato quando si ridimensiona anche il rigo. Maggiori informazioni in `<undefined>` [Setting the staff size], pagina `<undefined>`.

### *Ridimensionare oggetti della formattazione individualmente*

Un singolo oggetto della formattazione può essere ridimensionato coi comandi \code{tweak} o \code{override} per regolare la sua proprietà `font-size`:

```
\relative {
  % ridimensiona una testa di nota
  <f' \tweak font-size -4 b e>-5
  % ridimensiona una ditekgiatura
  bes-\tweak font-size 0 -3
```

```
% ridimensiona un'alterazione
\once \override Accidental.font-size = -4 bes!-^
% ridimensiona un'articolazione
\once \override Script.font-size = 4 bes!-^
}
```



Il valore predefinito di `font-size` per ogni oggetto della formattazione è elencato nella Guida al funzionamento interno. La proprietà `font-size` può essere impostata solo per quegli oggetti che supportano l'interfaccia di formattazione `font-interface`. Se `font-size` non è specificato nella lista 'Standard settings' dell'oggetto, il suo valore è 0. Si veda Sezione "All layout objects" in *Guida al Funzionamento Interno*.

### Capire la proprietà `fontSize`

La proprietà di contesto `fontSize` regola la dimensione relativa di tutti gli elementi della notazione basati su un glifo in un contesto:

```
\relative {
  \time 3/4
  d''4---5 c8( b a g) |
  \set fontSize = -6
  e'4-- c!8-4( b a g) |
  \set fontSize = 0
  fis4---3 e8( d) fis4 |
  g2.
}
```



Il valore di `fontSize` è un numero che indica la dimensione relativa alla dimensione standard dell'altezza del rigo corrente. Il valore predefinito di `fontSize` è 0; aggiungendo 6 a qualsiasi valore di `fontSize` si raddoppia la dimensione dei glifi e togliendo 6 si dimezza. Ogni punto aumenta la dimensione di circa il 12%.

Dato che le unità logaritmiche della proprietà `font-size` non sono del tutto intuitive, viene fornita per comodità la funzione scheme `magnification->font-size`. Per esempio, per ridurre la notazione musicale al 75% della dimensione predefinita si usa:

```
\set fontSize = #(magnification->font-size 0.75)
```

La funzione scheme `magstep` fa l'opposto: converte un valore di `font-size` in un fattore di ingrandimento.

La proprietà `fontSize` avrà effetto soltanto sugli elementi della notazione che sono disegnati con glifi, come le teste di nota, le alterazioni, i segni, etc. Non modificherà la dimensione del rigo stesso né ridimensionerà proporzionalmente gambi, travature o la spaziatura orizzontale. Per ridimensionare gambi, travature e spaziatura orizzontale insieme alla dimensione degli elementi della notazione (senza cambiare la dimensione del rigo), si usa il comando `\magnifyMusic` presentato prima. Per ridimensionare tutto, compreso il rigo, leggere `<undefined>` [Setting the staff size], pagina `<undefined>`.

Ogni volta che la *proprietà di contesto* `fontSize` è impostata, il suo valore viene aggiunto al valore della *proprietà del grob* `font-size` per i singoli oggetti di formattazione, prima che siano stampati i glifi. Ciò può creare confusione quando si impostano individualmente le proprietà `font-size` mentre è impostato anche `fontSize`:

```
% il valore predefinito di font-size per NoteHead è 0
% il valore predefinito di font-size per Fingering è -5
c''4-3
```

```
\set fontSize = -3
% la dimensione effettiva per NoteHead è ora -3
% la dimensione effettiva per Fingering è ora -8
c''4-3
```

```
\override Fingering.font-size = 0
% la dimensione effettiva per Fingering è ora -3
c''4-3
```



Sono anche disponibili le seguenti scorciatoie:

Comando	Equivalente a	Dimensione relativa
<code>\teeny</code>	<code>\set fontSize = -3</code>	71%
<code>\tiny</code>	<code>\set fontSize = -2</code>	79%
<code>\small</code>	<code>\set fontSize = -1</code>	89%
<code>\normalsize</code>	<code>\set fontSize = 0</code>	100%
<code>\large</code>	<code>\set fontSize = 1</code>	112%
<code>\huge</code>	<code>\set fontSize = 2</code>	126%
<code>\relative c'' {</code>		
<code>\teeny</code>		
<code>c4.-&gt; d8---3</code>		
<code>\tiny</code>		
<code>c4.-&gt; d8---3</code>		
<code>\small</code>		
<code>c4.-&gt; d8---3</code>		
<code>\normalsize</code>		
<code>c4.-&gt; d8---3</code>		
<code>\large</code>		
<code>c4.-&gt; d8---3</code>		
<code>\huge</code>		
<code>c4.-&gt; d8---3</code>		
<code>}</code>		



La modifica della dimensione del tipo di carattere si ottiene ridimensionando la dimensione, tra quelle predefinite, più vicina a quella desiderata. La dimensione standard (per `font-size = 0`) dipende dall'altezza standard del rigo: per un rigo di 20pt, viene scelto un tipo di carattere di 11pt.



## Comandi predefiniti

`\magnifyMusic`, `\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`.

## Vedi anche

Guida alla notazione: [\[Selecting font and font size\]](#), pagina [\[Setting the staff size\]](#), pagina [\[Formatting cue notes\]](#), pagina [\[Ossia staves\]](#).

File installati: `ly/music-functions-init.ly`, `ly/property-init.ly`.

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “font-interface” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Attualmente ci sono due bug che impediscono una corretta spaziatura orizzontale quando si usa `\magnifyMusic`. C'è un solo modo per eludere questi bug e non funziona in tutte le circostanze. Nell'esempio seguente, sostituire la variabile `mag` con un valore a piacere. Si può provare anche a togliere uno o entrambi i comandi `\newSpacingSection` e/o i comandi `\override` e `\revert`:

```
\magnifyMusic mag {
  \newSpacingSection
  \override Score.SpacingSpanner.spacing-increment = #(* 1.2 mag)
  [music]
  \newSpacingSection
  \revert Score.SpacingSpanner.spacing-increment
}
```

## Indicazioni di diteggiatura

Le indicazioni di diteggiatura si inseriscono con ‘*nota-numero*’:

```
\relative { c''4-1 d-2 f-4 e-3 }
```



Si può usare il testo incluso dentro `\markup` o tra virgolette per indicare un cambio di dito.

```
\relative {
  c''4-1 d-2 f\finger \markup \tied-lyric #"4~3" c\finger "2 - 3"
}
```



Si può aggiungere il simbolo del pollice per indicare che una nota deve essere suonata col pollice (ad esempio, nella musica per violoncello).

```
\relative { <a'_\thumb a'-3>2 <b'_\thumb b'-3> }
```



È possibile indicare la diteggiatura di ogni singola nota di un accordo specificandola dopo ciascuna altezza.

```
\relative {
  <c'-1 e-2 g-3 b-5>2 <d-1 f-2 a-3 c-5>
}
```



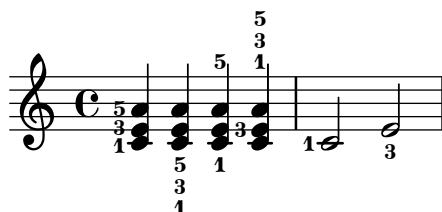
Le indicazioni di diteggiatura possono essere poste sopra o sotto il rigo, come è spiegato in Sezione 5.4.2 [Direction and placement], pagina 611.

## Frammenti di codice selezionati

*Controllare il posizionamento delle diteggiature di un accordo*

Il posizionamento dei numeri della diteggiatura può essere regolato in modo preciso. Perché l'orientamento funzioni, occorre usare il costrutto per gli accordi <> anche per una nota singola.

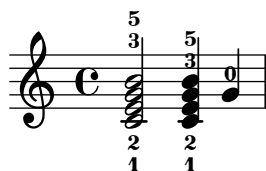
```
\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down right up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(left)
  <c-1>2
  \set fingeringOrientations = #'(down)
  <e-3>2
}
```



*Far sì che la diteggiatura appaia dentro il rigo*

Per impostazione predefinita, le diteggiature orientate verticalmente sono poste fuori dal rigo. Tuttavia, questo comportamento può essere annullato. Attenzione: bisogna usare il costrutto per gli accordi <>, anche se si riferisce a una singola nota.

```
\relative c' {
  <c-1 e-2 g-3 b-5>2
  \override Fingering.staff-padding = #'()
  <c-1 e-2 g-3 b-5>4 <g'-0>
}
```



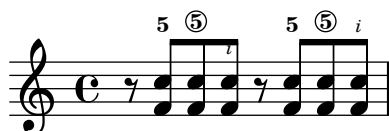
*Evitare le collisioni con le diteggiature degli accordi*

Diteggiature e numeri di corda applicati a note individuali evitano automaticamente le travature e i gambi, ma questo non vale per diteggiature e numeri di corda applicati alle singole note di un accordo. L'esempio seguente mostra come aggirare questo comportamento predefinito.

```
\relative c' {
  \set fingeringOrientations = #'(up)
  \set stringNumberOrientations = #'(up)
  \set strokeFingerOrientations = #'(up)

  % Default behavior
  r8
  <f c'-5>8
  <f c'\5>8
  <f c'-\rightHandFinger #2 >8

  % No tweak needed
  r8
  <f c'-5>8
  <f c'\5>8
  % Corrected to avoid collisions
  \override StrokeFinger.add-stem-support = ##t
  <f c'-\rightHandFinger #2 >8
}
```



## Vedi anche

Guida alla notazione: Sezione 5.4.2 [Direction and placement], pagina 611.

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “FingeringEvent” in *Guida al Funzionamento Interno*, Sezione “fingering-event” in *Guida al Funzionamento Interno*, Sezione “Fingering-engraver” in *Guida al Funzionamento Interno*, Sezione “New-fingering-engraver” in *Guida al Funzionamento Interno*, Sezione “Fingering” in *Guida al Funzionamento Interno*.

## Note nascoste

Le note nascoste (o invisibili o trasparenti) possono essere utili nella preparazione di esercizi di teoria e composizione.

```
\relative {
  c''4 d
  \hideNotes
  e4 f
  \unHideNotes
  g a
  \hideNotes
```

```

b
\unHideNotes
c
}

```



Questo comando rende invisibili le teste, i gambi e le code delle note, e le pause. Le travature sono invisibili se iniziano su una nota nascosta. Mentre gli oggetti attaccati a note invisibili sono comunque visibili.

```

\relative c'' {
  e8(\p f g a)--
  \hideNotes
  e8(\p f g a)--
}

```



## Comandi predefiniti

`\hideNotes`, `\unHideNotes`.

## Vedi anche

Manuale d'apprendimento: Sezione “Visibilità e colore degli oggetti” in *Manuale di Apprendimento*.

Guida alla notazione: `<undefined>` [Invisible rests], pagina `<undefined>`, Sezione 5.4.7 [Visibility of objects], pagina 618, `<undefined>` [Hiding staves], pagina `<undefined>`.

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Note-spacing-engraver” in *Guida al Funzionamento Interno*, Sezione “NoteSpacing” in *Guida al Funzionamento Interno*.

## Colorare gli oggetti

Si possono assegnare dei colori a ciascun oggetto. I nomi dei colori validi sono elencati nella `<undefined>` [List of colors], pagina `<undefined>`.

```

\override NoteHead.color = #red
c''4 c''
\override NoteHead.color = #(x11-color 'LimeGreen)
d''
\override Stem.color = #blue
e''

```



Si può accedere all'intera gamma di colori definita per X11 con la funzione Scheme `x11-color`. La funzione prende un argomento, che può essere un simbolo nella forma `'FooBar` o una stringa

nella forma "*FooBar*". La prima forma è più veloce da scrivere e più efficiente. Tuttavia, la seconda forma permette di accedere ai colori X11 attraverso la forma del nome che ha più di una parola.

La funzione `x11-color`, se non riesce a comprendere il parametro, restituisce il colore nero.

```
\relative c'' {
  \override Staff.StaffSymbol.color = #(x11-color 'SlateBlue2)
  \set Staff.instrumentName = \markup {
    \with-color #(x11-color 'navy) "Clarinet"
  }

  gis8 a
  \override Beam.color = #(x11-color "medium turquoise")
  gis a
  \override Accidental.color = #(x11-color 'DarkRed)
  gis a
  \override NoteHead.color = #(x11-color "LimeGreen")
  gis a
  % questo parametro è volutamente assurdo; notare che i gambi restano neri
  \override Stem.color = #(x11-color 'Boggle)
  b2 cis
}
```



I colori RGB esatti si specificano con la funzione Scheme `rgb-color`.

```
\relative c'' {
  \override Staff.StaffSymbol.color = #(x11-color 'SlateBlue2)
  \set Staff.instrumentName = \markup {
    \with-color #(x11-color 'navy) "Clarinet"
  }

  \override Stem.color = #(rgb-color 0 0 0)
  gis8 a
  \override Stem.color = #(rgb-color 1 1 1)
  gis8 a
  \override Stem.color = #(rgb-color 0 0 0.5)
  gis4 a
}
```



## Vedi anche

Guida alla notazione: [\[List of colors\]](#), pagina [\[The tweak command\]](#), pagina [\[The tweak command\]](#).

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

## Problemi noti e avvertimenti

Un colore X11 non ha necessariamente la stessa identica tonalità di un normale colore dal nome simile.

Non tutti i colori X11 sono distinguibili in un browser web. Per esempio, un browser potrebbe non mostrare alcuna differenza tra **LimeGreen** e **ForestGreen**. Per il web si consiglia di usare i colori normali (ovvero **blue**, **green**, **red**).

Le note in un accordo non possono essere colorate separatamente con un `\override`; al suo posto si usa `\tweak` o l'equivalente `\single\override`, vedi [\[The tweak command\]](#), pagina [\[undefined\]](#).

## Parentesi

Gli oggetti possono essere messi tra parentesi se si usa il comando `\parenthesize` prima dell'evento musicale. Se precede un accordo, viene messa tra parentesi ogni nota dell'accordo. Si possono mettere tra parentesi anche singole note di un accordo.

```
\relative {
  c''2 \parenthesize d
  c2 \parenthesize <c e g>
  c2 <c \parenthesize e g>
}
```



Si possono mettere tra parentesi anche oggetti diversi dalle note. Per le articolazioni è necessario usare un trattino prima del comando `\parenthesize`.

```
\relative {
  c''2-\parenthesize -. d
  c2 \parenthesize r
}
```



## Vedi anche

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Parenthesis-engraver” in *Guida al Funzionamento Interno*, Sezione “ParenthesesItem” in *Guida al Funzionamento Interno*, Sezione “parentheses-interface” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Se si mette tra parentesi un accordo, viene creata una parentesi per ogni nota dell'accordo invece di una sola grande parentesi per l'intero accordo.

## Gambi

Per ogni nota viene creato automaticamente un oggetto **Stem** (gambo). Vale anche per le semi-brevi e le pause, anche se i loro gambi sono resi invisibili.

I gambi si possono posizionare sopra o sotto, vedi Sezione 5.4.2 [Direction and placement], pagina 611.

## Comandi predefiniti

`\stemUp`, `\stemDown`, `\stemNeutral`.

## Frammenti di codice selezionati

*Direzione predefinita dei gambi sulla linea centrale del rigo*

La direzione predefinita dei gambi sulla linea centrale del rigo si imposta con la proprietà `neutral-direction` dell'oggetto `Stem`.

```
\relative c' ' {
  a4 b c b
  \override Stem.neutral-direction = #up
  a4 b c b
  \override Stem.neutral-direction = #down
  a4 b c b
}
```



*Cambiare automaticamente la direzione del gambo della nota centrale in base alla melodia*

LilyPond può modificare la direzione del gambo della nota centrale di un rigo in modo che segua la melodia: occorre aggiungere l'incisore `Melody_engraver` al contesto `Voice` e sovrascrivere la proprietà `neutral-direction` di `Stem`.

```
\relative c' ' {
  \time 3/4
  a8 b g f b g |
  c b d c b c |
}

\layout {
  \context {
    \Voice
    \consists "Melody_engraver"
    \autoBeamOff
    \override Stem.neutral-direction = #'()
  }
}
```



## Vedi anche

Guida alla notazione: Sezione 5.4.2 [Direction and placement], pagina 611.

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Stem\_engraver” in *Guida al Funzionamento Interno*, Sezione “Stem” in *Guida al Funzionamento Interno*, Sezione “stem-interface” in *Guida al Funzionamento Interno*.

### 1.7.2 Esterne al rigo

Questa sezione spiega come dare risalto agli elementi nel rigo attraverso delle note esterne al rigo.

#### Nuvoletta di aiuto

Si possono contrassegnare e nominare gli elementi della notazione tramite una nuvoletta quadrata. La sua funzione principale è spiegare la notazione.

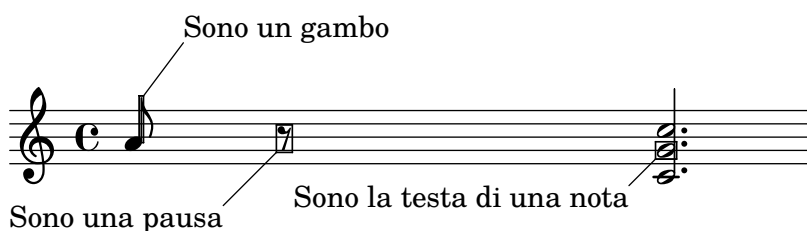
```
\relative c' ' {
  \new Voice \with { \consists "Balloon_engraver" }
  {
    \balloonGrobText #'Stem #'(3 . 4) \markup { "Sono un gambo" }
    a8
    \balloonGrobText #'Rest #'(-4 . -4) \markup { "Sono una pausa" }
    r
    <c, g'-\balloonText #'(-2 . -2) \markup { "Sono la testa di una nota" } c>2.
  }
}
```



Ci sono due funzioni musicali, `balloonGrobText` e `balloonText`; la prima si usa nella forma `\once \override` per attaccare del testo a un qualsiasi oggetto grafico (grob), mentre la seconda viene usata come `\tweak`, solitamente all'interno degli accordi, per attaccare del testo a una singola nota.

Il testo nella nuvoletta influenza la spaziatura delle note, ma è possibile modificare questo comportamento:

```
\relative c' ' {
  \new Voice \with { \consists "Balloon_engraver" }
  {
    \balloonGrobText #'Stem #'(3 . 4) \markup { "Sono un gambo" }
    a8
    \balloonGrobText #'Rest #'(-4 . -4) \markup { "Sono una pausa" }
    r
    \balloonLengthOn
    <c, g'-\balloonText #'(-2 . -2) \markup { "Sono la testa di una nota" } c>2.
  }
}
```





## Comandi predefiniti

`\balloonLengthOn`, `\balloonLengthOff`.

## Vedi anche

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Balloon\_engraver” in *Guida al Funzionamento Interno*, Sezione “BalloonTextItem” in *Guida al Funzionamento Interno*, Sezione “balloon-interface” in *Guida al Funzionamento Interno*.

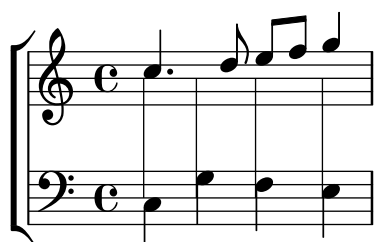
## Linee della griglia

Si possono disegnare delle linee verticali tra i righi sincronizzate con le note.

Si deve usare l'incisore `Grid_point_engraver` per creare le estremità delle linee, mentre l'incisore `Grid_line_span_engraver` serve a disegnare le linee. Per impostazione predefinita, le linee della griglia sono allineate orizzontalmente sotto e sul lato sinistro delle teste di nota. Le linee si estendono a partire dalle linee centrali di ciascun rigo. `gridInterval` deve specificare la durata che separa le linee.

```
\layout {
  \context {
    \Staff
    \consists "Grid_point_engraver"
    gridInterval = #(ly:make-moment 1/4)
  }
  \context {
    \Score
    \consists "Grid_line_span_engraver"
  }
}

\score {
  \new ChoirStaff <<
    \new Staff \relative {
      \stemUp
      c' '4. d8 e8 f g4
    }
    \new Staff \relative {
      \clef bass
      \stemDown
      c4 g' f e
    }
  >>
}
```



## Frammenti di codice selezionati

*Modificare l'aspetto delle linee della griglia*

L'aspetto delle linee della griglia può essere modificato sovrascrivendo alcune delle loro proprietà.

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \relative c'' {
        \stemUp
        c'4. d8 e8 f g4
      }
    }
    \new Staff {
      \relative c {
        % this moves them up one staff space from the default position
        \override Score.GridLine.extra-offset = #'(0.0 . 1.0)
        \stemDown
        \clef bass
        \once \override Score.GridLine.thickness = #5.0
        c4
        \once \override Score.GridLine.thickness = #1.0
        g'4
        \once \override Score.GridLine.thickness = #3.0
        f4
        \once \override Score.GridLine.thickness = #5.0
        e4
      }
    }
  >>
  \layout {
    \context {
      \Staff
      % set up grids
      \consists "Grid_point_engraver"
      % set the grid interval to one quarter note
      gridInterval = #(ly:make-moment 1/4)
    }
    \context {
      \Score
      \consists "Grid_line_span_engraver"
      % this moves them to the right half a staff space
      \override NoteColumn.X-offset = #-0.5
    }
  }
}
```



## Vedi anche

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Grid\_line\_span\_engraver” in *Guida al Funzionamento Interno*, Sezione “Grid\_point\_engraver” in *Guida al Funzionamento Interno*, Sezione “GridLine” in *Guida al Funzionamento Interno*, Sezione “GridPoint” in *Guida al Funzionamento Interno*, Sezione “grid-line-interface” in *Guida al Funzionamento Interno*, Sezione “grid-point-interface” in *Guida al Funzionamento Interno*.

## Parentesi analitiche

Nell’analisi musicale si usano le parentesi per indicare la struttura dei brani musicali. Sono supportate delle semplici parentesi orizzontali.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative {
  c''2\startGroup
  d\stopGroup
}
```



Le parentesi analitiche si possono annidare.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative {
  c''4\startGroup\startGroup
  d4\stopGroup
  e4\startGroup
  d4\stopGroup\stopGroup
}
```



## Vedi anche

Frammenti: Sezione “Editorial annotations” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Horizontal\_bracket\_engraver” in *Guida al Funzionamento Interno*, Sezione “HorizontalBracket” in *Guida al Funzionamento Interno*, Sezione “horizontal-bracket-interface” in *Guida al Funzionamento Interno*, Sezione “Staff” in *Guida al Funzionamento Interno*.

## 1.8 Testo

The image displays three examples of musical notation with text annotations in Italian. The first example shows a piano score in 3/4 time with notes and rests, annotated with *p con amabilità*, *ten.*, and *tranqu. dolce ten. ten.*. The second example shows a piano score with a melodic line and a bass line, annotated with *cantabile, con intimissimo sentimento, ma sempre molto dolce e semplice*, *non staccato*, and *molto p, sempre tranquillo ed egualmente, non rubato*. The third example shows a piano score with a melodic line and a bass line, annotated with *molto p, sempre tranquillo ed egualmente, non rubato*.

Questa sezione spiega come includere del testo (con vari tipi di formattazione) nelle partiture musicali.

Alcuni elementi testuali che non sono trattati qui sono discussi in altre sezioni specifiche: `<undefined>` [Vocal music], pagina `<undefined>`, `<undefined>` [Titles and headers], pagina `<undefined>`.

### 1.8.1 Inserimento del testo

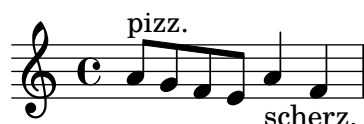
Questa sezione presenta vari modi di aggiungere del testo a una partitura.

**Nota:** Per scrivere caratteri accentati e speciali (come quelli di altre lingue), basta inserire semplicemente i caratteri nel file LilyPond, purché il file sia salvato in formato UTF-8. Ulteriori informazioni in [\[Text encoding\]](#), pagina [\[undefined\]](#).

#### Scritte

Si possono aggiungere a una partitura delle semplici indicazioni con del “testo tra virgolette”, come mostrato nell’esempio seguente. Tali indicazioni possono essere posizionate sopra o sotto il rigo, usando la sintassi descritta in Sezione 5.4.2 [\[Direction and placement\]](#), pagina 611.

```
\relative { a'8^"pizz." g f e a4-"scherz." f }
```



In realtà questa sintassi è una scorciatoia; si può specificare una formattazione del testo più complessa usando in modo esplicito un blocco `\markup`, come è spiegato in [\[Formatting text\]](#), pagina [\[undefined\]](#).

```
\relative {
  a'8^\markup { \italic pizz. } g f e
  a4_\markup { \tiny scherz. \bold molto } f }
```



Le indicazioni testuali, di norma, non influenzano la spaziatura delle note. Ma è possibile far sì che la loro larghezza venga presa in considerazione: nell’esempio seguente la prima stringa di testo non influenza la spaziatura, mentre la seconda sì.

```
\relative {
  a'8^"pizz." g f e
  \textLengthOn
  a4_"scherzando" f
}
```



Oltre alle scritte, si possono attaccare alle note anche le articolazioni. Ulteriori informazioni in [\[Articulations and ornamentations\]](#), pagina [\[undefined\]](#).

Per maggiori informazioni sull’ordinamento relativo delle scritte e delle articolazioni si veda Sezione “Posizionamento degli oggetti” in *Manuale di Apprendimento*.

#### Comandi predefiniti

```
\textLengthOn, \textLengthOff.
```

## Vedi anche

Manuale d'apprendimento: Sezione “Posizionamento degli oggetti” in *Manuale di Apprendimento*.

Guida alla notazione: [\[Formatting text\]](#), pagina [\[undefined\]](#), Sezione 5.4.2 [\[Direction and placement\]](#), pagina 611, [\[Articulations and ornamentations\]](#), pagina [\[undefined\]](#).

Frammenti: Sezione “Text” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TextScript” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

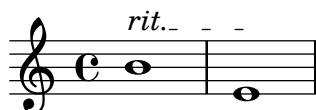
Per verificare che le scritte e il testo vocale siano entro i margini occorrono ulteriori calcoli. Nei casi in cui è richiesta un'esecuzione leggermente più veloce, usare

```
\override Score.PaperColumn.keep-inside-line = ##f
```

## Estensori del testo

Alcune indicazioni esecutive, per esempio *rallentando* o *accelerando*, appaiono in forma testuale e vengono estese lungo molteplici note con delle linee punteggiate. Tali oggetti, chiamati “estensori” (spanner), si creano collegando due note con la seguente sintassi:

```
\relative {
  \override TextSpanner.bound-details.left.text = "rit."
  b'1\startTextSpan
  e,\stopTextSpan
}
```



La stringa testuale da stampare viene impostata attraverso le proprietà dell'oggetto. Per impostazione predefinita, appare in corsivo, ma si può ottenere una formattazione diversa tramite i blocchi `\markup`, come è spiegato in [\[undefined\]](#) [\[Formatting text\]](#), pagina [\[undefined\]](#).

```
\relative {
  \override TextSpanner.bound-details.left.text =
    \markup { \upright "rit." }
  b'1\startTextSpan c
  e,\stopTextSpan
}
```



Lo stile della linea, così come la stringa testuale, può essere definito come una proprietà dell'oggetto. Questa sintassi è descritta in Sezione 5.4.8 [\[Line styles\]](#), pagina 625.

## Comandi predefiniti

```
\textSpannerUp, \textSpannerDown, \textSpannerNeutral.
```

## Problemi noti e avvertimenti

LilyPond è capace di gestire un solo estensore del testo per ogni voce.

## Frammenti di codice selezionati

### *Estensore testuale della dinamica personalizzato*

Si possono definire estensori testuali personalizzati che fanno uso delle forcine e dei crescendo testuali. `\<` e `\>` generano le forcine, `\cresc` etc. generano gli estensori testuali.

```
% Some sample text dynamic spanners, to be used as postfix operators
crpoco =
#(make-music 'CrescendoEvent
    'span-direction START
    'span-type 'text
    'span-text "cresc. poco a poco")

\relative c' {
  c4\cresc d4 e4 f4 |
  g4 a4\! b4\crpoco c4 |
  c4 d4 e4 f4 |
  g4 a4\! b4\< c4 |
  g4\dim a4 b4\decreasc c4\!
}
```



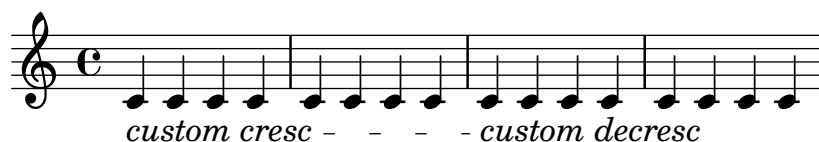
### *Estensore testuale della dinamica personalizzato*

Funzioni postfix per estensori testuali personalizzati del crescendo. Gli estensori devono iniziare sulla prima nota della misura; e bisogna usare `-\mycresc`, altrimenti l'inizio dell'estensore viene assegnato alla nota successiva.

```
% Two functions for (de)crescendo spanners where you can explicitly give the
% spanner text.
mycresc =
#(define-music-function (mymarkup) (markup?)
  (make-music 'CrescendoEvent
    'span-direction START
    'span-type 'text
    'span-text mymarkup))

mydecreasc =
#(define-music-function (mymarkup) (markup?)
  (make-music 'DecrescendoEvent
    'span-direction START
    'span-type 'text
    'span-text mymarkup))

\relative c' {
  c4-\mycresc "custom cresc" c4 c4 c4 |
  c4 c4 c4 c4 |
  c4-\mydecreasc "custom decreasc" c4 c4 c4 |
  c4 c4\! c4 c4
}
```



## Vedi anche

Guida alla notazione: Sezione 5.4.8 [Line styles], pagina 625, [\[Dynamics\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [Formatting text], pagina [\[undefined\]](#).

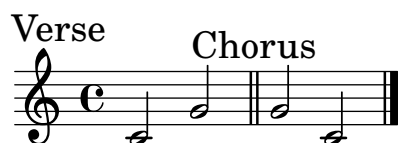
Frammenti: Sezione “Text” in *Frammenti di codice*, Sezione “Expressive marks” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TextSpanner” in *Guida al Funzionamento Interno*.

## Indicazioni testuali

Si possono aggiungere vari elementi testuali a una partitura tramite la sintassi descritta in [\[undefined\]](#) [Rehearsal marks], pagina [\[undefined\]](#),:

```
\relative {
  \mark "Verse"
  c'2 g'
  \bar "||"
  \mark "Chorus"
  g2 c,
  \bar "|."
}
```



Questa sintassi permette di porre del testo sopra una stanghetta; una formattazione del testo più complessa è possibile grazie al blocco `\markup`, come è spiegato in [\[undefined\]](#) [Formatting text], pagina [\[undefined\]](#),:

```
\relative {
  <c' e>1
  \mark \markup { \italic { colla parte } }
  <d f>2 <e g>
  <c f aes>1
}
```



Questa sintassi permette anche di stampare segni speciali, come coda, segno o corona, se si specifica il nome appropriato del simbolo, come è spiegato in [\[undefined\]](#) [Music notation inside markup], pagina [\[undefined\]](#),:

```
\relative {
  <bes' f>2 <aes d>
  \mark \markup { \musicglyph #"scripts.ufermata" }
  <e g>1
}
```





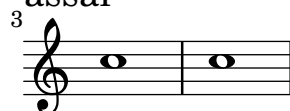
Tali oggetti vengono posizionati soltanto sopra il rigo superiore della partitura; a seconda che siano inseriti alla fine o a metà di una battuta, possono trovarsi sopra la stanghetta o tra le note. Se inserito prima di un'interruzione di linea, l'indicazione apparirà all'inizio della linea successiva.

```
\relative c'' {
  \mark "Allegro"
  c1 c
  \mark "assai" \break
  c c
}
```

### Allegro



### assai



## Comandi predefiniti

`\markLengthOn`, `\markLengthOff`.

## Frammenti di codice selezionati

*Posizionare le indicazioni alla fine di una linea*

È possibile posizionare le indicazioni alla fine della linea corrente, invece che all'inizio della linea successiva. In tali casi, può essere preferibile allineare l'estremità destra dell'indicazione alla stanghetta.

```
\relative c'' {
  g2 c
  d,2 a'
  \once \override Score.RehearsalMark.break-visibility = #end-of-line-visible
  \once \override Score.RehearsalMark.self-alignment-X = #RIGHT
  \mark "D.C. al Fine"
  \break
  g2 b,
  c1 \bar "||"
}
```

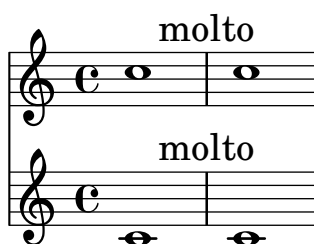
### D.C. al Fine



*Stampare le indicazioni su ogni rigo*

Sebbene le indicazioni testuali siano di norma collocate solo sopra il rigo più alto, è possibile farle apparire su ogni rigo.

```
\score {
  <<
    \new Staff { c''1 \mark "molto" c'' }
    \new Staff { c'1 \mark "molto" c' }
  >>
  \layout {
    \context {
      \Score
      \remove "Mark_engraver"
      \remove "Staff_collecting_engraver"
    }
    \context {
      \Staff
      \consists "Mark_engraver"
      \consists "Staff_collecting_engraver"
    }
  }
}
```



## Vedi anche

Guida alla notazione: [\[Rehearsal marks\]](#), pagina [\[Formatting text\]](#), pagina [\[Music notation inside markup\]](#), pagina [\[The Feta font\]](#), pagina [\[The Feta font\]](#).

Frammenti: Sezione “Text” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “MarkEvent” in *Guida al Funzionamento Interno*, Sezione “Mark\_engraver” in *Guida al Funzionamento Interno*, Sezione “RehearsalMark” in *Guida al Funzionamento Interno*.

## Testo separato

Un blocco `\markup` può esistere di per sé, fuori da qualsiasi blocco `\score`, come un’ “espressione di livello superiore”. Questa sintassi è descritta in [\[File structure\]](#), pagina [\[File structure\]](#).

```
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
```

Tomorrow, and tomorrow, and tomorrow...

Ciò permette di stampare il testo in modo autonomo dalla musica, ed è utile soprattutto quando il file di input contiene vari brani musicali, come è spiegato in [\[Multiple scores in a book\]](#), pagina [\[undefined\]](#).

```
\score {
  c'1
}
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
\score {
  c'1
}
```



Tomorrow, and tomorrow, and tomorrow...



Blocchi di testo separati possono essere estesi per molte pagine, rendendo possibile la realizzazione di documenti o libri interamente con LilyPond. Questa funzionalità, e la sintassi specifica che richiede, è descritta in [\[Multi-page markup\]](#), pagina [\[undefined\]](#).

## Comandi predefiniti

`\markup`, `\markuplist`.

## Frammenti di codice selezionati

*Testo separato su due colonne*

Il testo separato può essere disposto su varie colonne con i comandi di `\markup`:

```
\markup {
  \fill-line {
    \hspace #1
    \column {
      \line { 0 sacrum convivium }
      \line { in quo Christus sumitur, }
      \line { recolitur memoria passionis ejus, }
      \line { mens impletur gratia, }
      \line { futurae gloriae nobis pignus datur. }
      \line { Amen. }
    }
  }
  \hspace #2
  \column \italic {
    \line { 0 sacred feast }
    \line { in which Christ is received, }
    \line { the memory of His Passion is renewed, }
    \line { the mind is filled with grace, }
```

```

\line { and a pledge of future glory is given to us. }
\line { Amen. }
}
\hspace #1
}
}

```

O sacrum convivium	<i>O sacred feast</i>
in quo Christus sumitur,	<i>in which Christ is received,</i>
recolitur memoria passionis ejus,	<i>the memory of His Passion is renewed,</i>
mens impletur gratia,	<i>the mind is filled with grace,</i>
futurae gloriae nobis pignus datur.	<i>and a pledge of future glory is given to us.</i>
Amen.	<i>Amen.</i>

## Vedi anche

Guida alla notazione: [\[Formatting text\]](#), pagina [\[undefined\]](#), [\[File structure\]](#), pagina [\[undefined\]](#), [\[Multiple scores in a book\]](#), pagina [\[undefined\]](#), [\[Multi-page markup\]](#), pagina [\[undefined\]](#).

Frammenti: Sezione “Text” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TextScript” in *Guida al Funzionamento Interno*.

## 1.8.2 Formattazione del testo

Questa sezione presenta la formattazione del testo basilare e quella avanzata, usando la sintassi specifica della modalità `\markup`.

### Introduzione al testo a margine

Un blocco `\markup` permette di comporre del testo con un’ampia sintassi chiamata “modalità markup”.

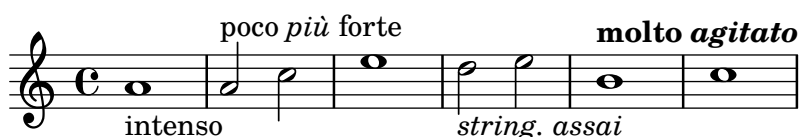
La sintassi di markup è simile alla solita sintassi di LilyPond: un’espressione `\markup` viene racchiusa tra parentesi graffe `{...}`. Una singola parola viene considerata un’espressione minima, e quindi non è necessario racchiuderla tra parentesi.

Diversamente dalle indicazioni testuali “tra virgolette”, i blocchi `\markup` possono contenere espressioni o comandi di markup annidati, inseriti col carattere di barra inversa `\`. Tali comandi hanno effetto solo sulla prima espressione che segue.

```

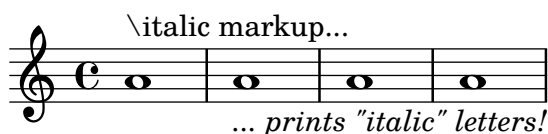
\relative {
  a'1-\markup intenso
  a2^\markup { poco \italic più forte }
  c e1
  d2_\markup { \italic "string. assai" }
  e
  b1^\markup { \bold { molto \italic agitato } }
  c
}

```



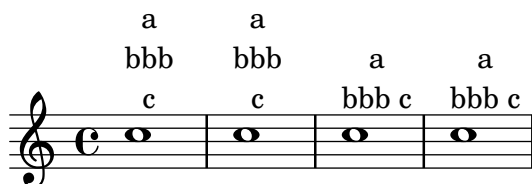
Un blocco `\markup` può contenere anche stringhe di testo tra virgolette. Tali stringhe vengono trattate come espressioni testuali minime, e quindi qualsiasi comando di markup o carattere speciale (come `\` e `#`) apparirà alla lettera senza influenzare la formattazione del testo. Le stesse doppie virgolette possono essere stampate facendole precedere da una barra inversa.

```
\relative {
  a'1^"\italic markup..."
  a_\markup { \italic "... prints \"italic\" letters!" }
  a a
}
```



Perché sia trattata come un'espressione distinta, una lista di parole deve essere racchiusa tra virgolette doppie o preceduta da un comando. Il modo in cui le espressioni musicali sono definite influenza il modo in cui saranno sistemate, centrate e allineate; nell'esempio seguente, la seconda espressione di `\markup` viene trattata nello stesso modo della prima:

```
\relative c'' {
  c1^\markup { \center-column { a bbb c } }
  c1^\markup { \center-column { a { bbb c } } }
  c1^\markup { \center-column { a \line { bbb c } } }
  c1^\markup { \center-column { a "bbb c" } }
}
```



I markup possono essere salvati in delle variabili, che possono poi essere attaccate direttamente alle note:

```
allegro = \markup { \bold \large Allegro }
```

```
{
  d''8.^allegro
  d'16 d'4 r2
}
```



Una lista completa dei comandi specifici di `\markup` si trova in [\[Text markup commands\]](#), pagina [\[Text markup commands\]](#).

## Vedi anche

Guida alla notazione: [\[Text markup commands\]](#), pagina [\[Text markup commands\]](#).

Frammenti: Sezione “Text” in *Frammenti di codice*.

File installati: `scm/markup.scm`.

## Problemi noti e avvertimenti

Gli errori di sintassi relativi alla modalità markup possono essere poco chiari.

## Scelta del tipo di carattere e della dimensione

La modalità markup permette di cambiare il tipo di carattere:

```
\relative {
  d''1^\markup {
    \bold { Più mosso }
    \italic { non troppo \underline Vivo }
  }
  r2 r4 r8
  d,_\markup { \italic quasi \smallCaps Tromba }
  f1 d2 r
}
```



Si può modificare la dimensione del tipo di carattere, rispetto alla dimensione globale del rigo, in vari modi.

Si può impostare su una dimensione predefinita,

```
\relative b' {
  b1_\markup { \huge Sinfonia }
  b1^\markup { \teeny da }
  b1-\markup { \normalsize camera }
}
```



oppure in modo proporzionale rispetto al valore precedente,

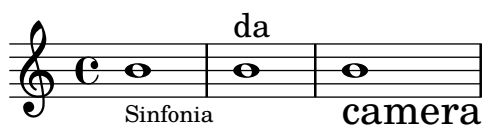
```
\relative b' {
  b1_\markup { \larger Sinfonia }
  b1^\markup { \smaller da }
  b1-\markup { \magnify #0.6 camera }
}
```



Può essere aumentata o diminuita rispetto al valore impostato per la dimensione globale del rigo:

```
\relative b' {
  b1_\markup { \fontsize #-2 Sinfonia }
  b1^\markup { \fontsize #1 da }
  b1-\markup { \fontsize #3 camera }
}
```

}



Si può impostare anche su una dimensione fissa (in punti), indipendentemente dalla dimensione globale del rigo:

```
\relative b' {
  b1_\markup { \abs-fontsize #20 Sinfonia }
  b1^\markup { \abs-fontsize #8 da }
  b1-\markup { \abs-fontsize #14 camera }
}
```



Se il testo contiene degli spazi, è meglio racchiuderlo tutto tra virgolette, in modo che la dimensione di ciascun spazio sia adatta alla dimensione degli altri caratteri.

```
\markup \fontsize #6 \bold { Sinfonia da camera }
\markup \fontsize #6 \bold { "Sinfonia da camera" }
```

## Sinfonia da camera

## Sinfonia da camera

È possibile stampare il testo come pedice o apice. Per impostazione predefinita, questo appaiono in corpo più piccolo, ma si può usare anche un corpo normale:

```
\markup {
  \column {
    \line { 1 \super st movement }
    \line { 1 \normal-size-super st movement }
    \sub { (part two) } }
}
```

1<sup>st</sup> movement  
1<sup>st</sup> movement<sub>(part two)</sub>

La modalità di markup fornisce un modo semplice per scegliere famiglie di caratteri diverse. Se non specificato altrimenti, viene scelto automaticamente il carattere tipografico con grazie (il tipo romano); nell'ultima linea dell'esempio seguente non c'è differenza tra la prima e la seconda parola.

```
\markup {
  \column {
    \line { Act \number 1 }
    \line { \sans { Scene I. } }
    \line { \typewriter { Verona. An open place. } }
```

```

\line { Enter \roman Valentine and Proteus. }
}
}

```

## Act 1

### Scene I.

Verona. An open place.

**Enter Valentine and Proteus.**

Alcune di queste famiglie di caratteri, usate per elementi specifici come i numeri o le dinamiche, non forniscono tutti i caratteri, come accennato in [\[New dynamic marks\]](#), pagina [\[undefined\]](#), e [\[Manual repeat marks\]](#), pagina [\[undefined\]](#).

Se usati all'interno di una parola, alcuni comandi che cambiano il tipo di carattere o la formattazione potrebbero produrre uno spazio vuoto indesiderato. Si può facilmente risolvere concatenando insieme gli elementi testuali:

```

\markup {
  \column {
    \line {
      \concat { 1 \super st }
      movement
    }
    \line {
      \concat { \dynamic p , }
      \italic { con dolce espressione }
    }
  }
}

```

1<sup>st</sup> movement

***p***, con dolce espressione

Una lista completa dei comandi per cambiare il tipo di carattere o per usare tipi di carattere personalizzati si trova in Sezione A.11.1 [\[Font\]](#), pagina 683.

È possibile anche definire i propri gruppi di tipi di carattere, come è spiegato in [\[undefined\]](#) [\[Fonts\]](#), pagina [\[undefined\]](#).

## Comandi predefiniti

`\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`, `\smaller`, `\larger`.

## Vedi anche

Guida alla notazione: Sezione A.11.1 [\[Font\]](#), pagina 683, [\[undefined\]](#) [\[New dynamic marks\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Manual repeat marks\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Fonts\]](#), pagina [\[undefined\]](#).

File installati: `scm/define-markup-commands.scm`.

Frammenti: Sezione “Text” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TextScript” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

L'uso dei comandi di dimensionamento dei caratteri `\teeny`, `\tiny`, `\small`, `\normalsize`, `\large` e `\huge` produce una spaziatura della linea imprevedibile rispetto all'uso di `\fontsize`.

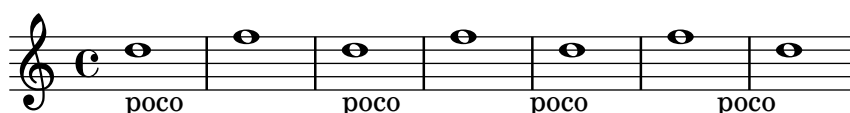


## Allineamento del testo

Questa sottosezione spiega come posizionare il testo nella modalità markup. Gli oggetti markup possono anche essere spostati interamente tramite la sintassi descritta in Sezione “Spostare gli oggetti” in *Manuale di Apprendimento*.

Gli oggetti di markup possono essere allineati in vari modi. Per impostazione predefinita, l’indicazione testuale è allineata rispetto al suo margine sinistro: nell’esempio seguente, non c’è differenza tra il primo e il secondo markup.

```
\relative {
  d''1-\markup { poco }
  f
  d-\markup { \left-align poco }
  f
  d-\markup { \center-align { poco } }
  f
  d-\markup { \right-align poco }
}
```



L’allineamento orizzontale può essere ritoccato usando un valore numerico:

```
\relative {
  a'1-\markup { \halign #-1 poco }
  e'
  a,-\markup { \halign #0 poco }
  e'
  a,-\markup { \halign #0.5 poco }
  e'
  a,-\markup { \halign #2 poco }
}
```



Alcuni oggetti possono avere proprie procedure di allineamento, e dunque non sono influenzate da questi comandi. È possibile spostare tali oggetti di markup tutti insieme, come mostrato ad esempio in [\[Text marks\]](#), pagina [\[Text marks\]](#).

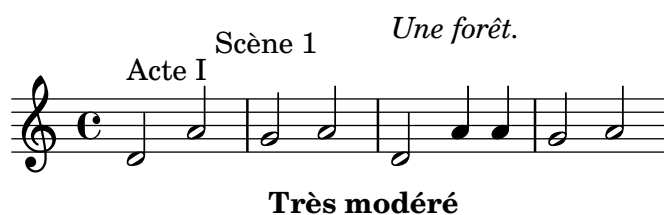
L’allineamento verticale è un po’ più complesso. Come si è detto prima, gli oggetti di markup possono essere spostati tutti insieme; tuttavia è anche possibile spostare elementi specifici all’interno di un blocco markup. In questo caso l’elemento da spostare deve essere preceduto da un *punto di riferimento*, che può essere un altro elemento markup o un oggetto invisibile. L’esempio seguente illustra queste due possibilità; l’ultimo markup in questo esempio non ha un punto di riferimento e di conseguenza non si muove.

```
\relative {
  d'2^\markup {
    Acte I
    \raise #2 { Scène 1 }
  }
}
```

```

a'
g_\markup {
  \null
  \lower #4 \bold { Très modéré }
}
a
d,^\markup {
  \raise #4 \italic { Une forêt. }
}
a'4 a g2 a
}

```

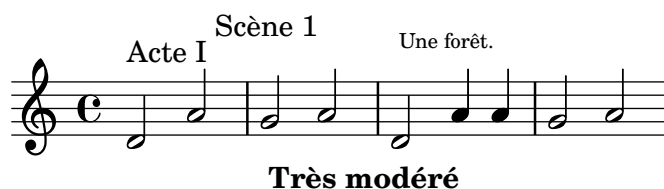


Alcuni comandi possono cambiare l'allineamento sia orizzontale che verticale degli oggetti testuali in modalità markup. Qualsiasi oggetto interessato da questi comandi deve essere preceduto da un punto di riferimento:

```

\relative {
  d'2^\markup {
    Acte I
    \translate #'(-1 . 2) "Scène 1"
  }
  a'
  g_\markup {
    \null
    \general-align #Y #3.2 \bold "Très modéré"
  }
  a
  d,^\markup {
    \null
    \translate-scaled #'(-1 . 2) \teeny "Une forêt."
  }
  a'4 a g2 a
}

```



Un oggetto markup può includere varie linee di testo. Nell'esempio seguente, ogni elemento o espressione viene posizionato sulla sua linea, allineato a sinistra o centrato:

```

\markup {
  \column {
    a
    "b c"
  }
}

```

```

\line { d e f }
}
\hspace #10
\center-column {
  a
  "b c"
  \line { d e f }
}
}

```

```

a          a
b c        b c
d e f      d e f

```

Analogoamente, una lista di elementi o espressioni può essere distesa per riempire l'intera larghezza orizzontale della linea (se c'è un solo elemento, verrà centrato sulla pagina). Queste espressioni possono a loro volta includere del testo multilinea o una qualsiasi altra espressione di markup:

```

\markup {
  \fill-line {
    \line { William S. Gilbert }
    \center-column {
      \huge \smallCaps "The Mikado"
      or
      \smallCaps "The Town of Titipu"
    }
    \line { Sir Arthur Sullivan }
  }
}
\markup {
  \fill-line { 1885 }
}

```

William S. Gilbert

THE MIKADO  
or  
THE TOWN OF TITIPU

Sir Arthur Sullivan

1885

Indicazioni testuali lunghe possono andare a capo automaticamente in base alla larghezza della linea specificata. Possono essere allineate a sinistra o giustificate, come mostra l'esempio seguente.

```

\markup {
  \column {
    \line \smallCaps { La vida breve }
    \line \bold { Acto I }
    \wordwrap \italic {
      (La escena representa el corral de una casa de
      gitanos en el Albaicín de Granada. Al fondo una
      puerta por la que se ve el negro interior de
      una Fragua, iluminado por los rojos resplandores
    }
  }
}

```

```

    del fuego.)
}
\hspace #0

\line \bold { Acto II }
\override #'(line-width . 50)
\justify \italic {
  (Calle de Granada. Fachada de la casa de Carmela
  y su hermano Manuel con grandes ventanas abiertas
  a través de las que se ve el patio
  donde se celebra una alegre fiesta)
}
}
}

```

LA VIDA BREVE

### **Acto I**

*(La escena representa el corral de una casa de gitanos en el Albaicín de Granada. Al fondo una puerta por la que se ve el negro interior de una Fragua, iluminado por los rojos resplandores del fuego.)*

### **Acto II**

*(Calle de Granada. Fachada de la casa de Carmela y su hermano Manuel con grandes ventanas abiertas a través de las que se ve el patio donde se celebra una alegre esta)*

Una lista completa dei comandi di allineamento del testo si trova in Sezione A.11.2 [Align], pagina 693.

## **Vedi anche**

Manuale d'apprendimento: Sezione "Spostare gli oggetti" in *Manuale di Apprendimento*.

Guida alla notazione: Sezione A.11.2 [Align], pagina 693, [\[Text marks\]](#), pagina [\[Text marks\]](#).

File installati: `scm/define-markup-commands.scm`.

Frammenti: Sezione "Text" in *Frammenti di codice*.

Guida al funzionamento interno: Sezione "TextScript" in *Guida al Funzionamento Interno*.

## **Notazione grafica nel blocco markup**

Si possono aggiungere vari oggetti grafici a una partitura attraverso i comandi di markup.

Alcuni comandi di markup consentono di decorare gli elementi testuali con degli elementi grafici, come è illustrato nell'esempio seguente.

```

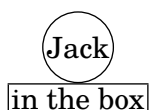
\markup \fill-line {
  \center-column {
    \circle Jack
    \box "in the box"
    \null
  }
  \line {
    Erik Satie
    \hspace #3
  }
}

```

```

    \bracket "1866 - 1925"
  }
  \null
  \rounded-box \bold Prelude
}
}

```



Erik Satie    [1866 - 1925]

**Prelude**

Alcuni comandi possono richiedere un aumento del padding intorno al testo; per farlo si usano dei comandi di markup, descritti in modo esaustivo in Sezione A.11.2 [Align], pagina 693.

```

\markup \fill-line {
  \center-column {
    \box "Charles Ives (1874 - 1954)"
    \null
    \box \pad-markup #2 "THE UNANSWERED QUESTION"
    \box \pad-x #8 "A Cosmic Landscape"
    \null
  }
}
\markup \column {
  \line {
    \hspace #10
    \box \pad-to-box #'(-5 . 20) #'(0 . 5)
    \bold "Largo to Presto"
  }
  \pad-around #3
  "String quartet keeps very even time,
  Flute quartet keeps very uneven time."
}

```

Charles Ives (1874 - 1954)

THE UNANSWERED QUESTION

A Cosmic Landscape

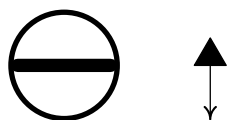
**Largo to Presto**

String quartet keeps very even time, Flute quartet keeps very uneven time.

Si possono produrre altri elementi grafici o simboli che non richiedono alcun testo. Come con qualsiasi espressione di markup, tali oggetti possono essere combinati.

```
\markup {
  \combine
    \draw-circle #4 #0.4 ##f
    \filled-box #'(-4 . 4) #'(-0.5 . 0.5) #1
  \hspace #5

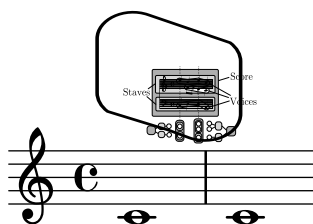
  \center-column {
    \triangle ##t
    \combine
      \draw-line #'(0 . 4)
      \arrow-head #Y #DOWN ##f
  }
}
```



Le funzionalità grafiche avanzate comprendono la possibilità di includere file di immagini convertite nel formato Encapsulated PostScript (*eps*), oppure di inserire la grafica direttamente nel file di input, usando del codice PostScript nativo. In tal caso, può essere utile specificare esplicitamente la dimensione del disegno, come è mostrato sotto:

```
c'1^\markup {
  \combine
    \epsfile #X #10 #"./context-example.eps"
    \with-dimensions #'(0 . 6) #'(0 . 10)
    \postscript #"
      -2 3 translate
      2.7 2 scale
      newpath
      2 -1 moveto
      4 -2 4 1 1 arct
      4 2 3 3 1 arct
      0 4 0 3 1 arct
      0 0 1 -1 1 arct
      closepath
      stroke"
}
```

c'



Una lista completa dei comandi specifici per la grafica si trova in Sezione A.11.3 [Graphic], pagina 708.

## Vedi anche

Guida alla notazione: Sezione A.11.2 [Align], pagina 693, Sezione 5.4.4 [Dimensions], pagina 613, [\[Editorial annotations\]](#), pagina [\[Graphic\]](#), pagina 708.

File installati: `scm/define-markup-commands.scm`, `scm/stencil.scm`.

Frammenti: Sezione “Text” in *Frammenti di codice*.

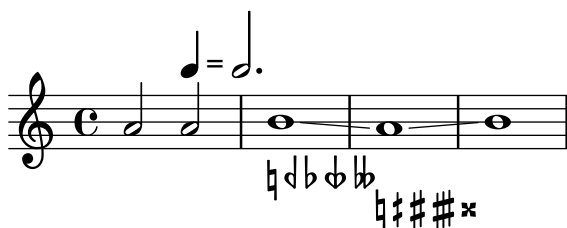
Guida al funzionamento interno: Sezione “TextScript” in *Guida al Funzionamento Interno*.

## Notazione musicale nel blocco markup

Si possono aggiungere vari elementi della notazione musicale dentro un oggetto markup.

Per le note e le alterazioni esistono dei comandi markup appositi:

```
a'2 a'^\markup {
  \note #"4" #1
  =
  \note-by-number #1 #1 #1.5
}
b'1_\markup {
  \natural \semiflat \flat
  \sesquiflat \doubleflat
}
\glissando
a'1_\markup {
  \natural \semisharp \sharp
  \sesquisharp \doublesharp
}
\glissando b'
```



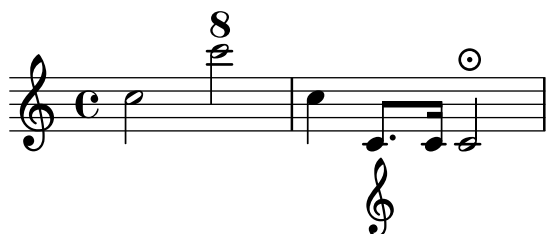
Anche altri oggetti della notazione possono essere stampati in modalità markup:

```
\relative {
  g1 bes
  ees\finger \markup \tied-lyric #"4~1"
  fis_\markup { \dynamic rf }
  bes^\markup {
    \beam #8 #0.1 #0.5
  }
  cis
  d-\markup {
    \markalphabet #8
    \markletter #8
  }
}
```



Più in generale, qualsiasi simbolo musicale disponibile può essere incluso separatamente in un oggetto markup, come è illustrato sotto. Una lista completa di questi simboli e dei loro nomi si trova in [\[The Feta font\]](#), pagina [\[The Feta font\]](#).

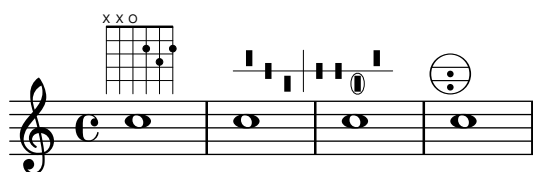
```
\relative {
  c''2
  c'^\markup { \musicglyph #"eight" }
  c,4
  c,8._\markup { \musicglyph #"clefs.G_change" }
  c16
  c2^\markup { \musicglyph #"timesig.neomensural94" }
}
```



Un altro modo per stampare glifi non testuali è descritto in [\[Fonts explained\]](#), pagina [\[Fonts explained\]](#). È utile per stampare parentesi di varie dimensioni.

La modalità markup supporta anche i diagrammi per strumenti specifici:

```
\relative {
  c''1^\markup {
    \fret-diagram-terse #"x;x;o;2;3;2;"
  }
  c^\markup {
    \harp-pedal #"^~v|--ov^"
  }
  c
  c^\markup {
    \combine
    \musicglyph #"accordion.discant"
    \combine
    \raise #0.5 \musicglyph #"accordion.dot"
    \raise #1.5 \musicglyph #"accordion.dot"
  }
}
```



Questi diagrammi sono documentati in Sezione A.11.5 [\[Instrument Specific Markup\]](#), pagina 722.

È possibile annidare perfino un'intera partitura in un oggetto markup. In tal caso, il blocco `\score` annidato deve contenere un blocco `\layout`, come è illustrato qui:

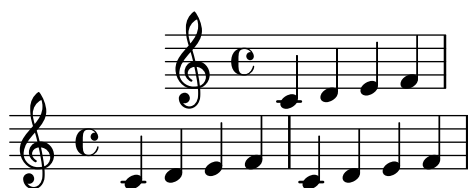
```
\relative {
```



```

c'4 d^\markup {
  \score {
    \relative { c'4 d e f }
    \layout { }
  }
}
e f |
c d e f
}

```



Una lista completa dei comandi relativi alla notazione musicale si trova in Sezione A.11.4 [Music], pagina 716.

## Vedi anche

Guida alla notazione: Sezione A.11.4 [Music], pagina 716, [\[The Feta font\]](#), pagina [\[Fonts explained\]](#), pagina [\[The Feta font\]](#).

File installati: `scm/define-markup-commands.scm`, `scm/fret-diagrams.scm`, `scm/harp-pedals.scm`.

Frammenti: Sezione “Text” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TextScript” in *Guida al Funzionamento Interno*.

## Testo formattato su più pagine

Sebbene gli oggetti di markup standard non possano avere interruzioni, una specifica sintassi permette di inserire linee di testo che possono estendersi per varie pagine:

```

\markuplist {
  \justified-lines {
    Un testo molto lungo di linee giustificate.
    ...
  }
  \wordwrap-lines {
    Un altro paragrafo molto lungo.
    ...
  }
  ...
}

```

Un testo molto lungo di linee giustificate. ...

Un altro paragrafo molto lungo. ...

...

Questa sintassi accetta una lista di oggetti di markup, che possono essere

- il risultato di un comando `\markuplist`,

- una lista di markup,
- una lista di `\markuplists`.

Una lista completa dei comandi che si possono usare con `\markuplist` si trova in `<undefined>` [Text markup list commands], pagina `<undefined>`.

## Vedi anche

Guida alla notazione: `<undefined>` [Text markup list commands], pagina `<undefined>`.

Estendere LilyPond: Sezione “New markup list command definition” in *Estendere*.

File installati: `scm/define-markup-commands.scm`.

Frammenti: Sezione “Text” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “TextScript” in *Guida al Funzionamento Interno*.

## Comandi predefiniti

`\markuplist`.

### 1.8.3 Tipi di carattere

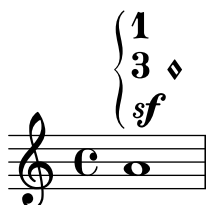
Questa sezione presenta il modo in cui sono gestiti i tipi di carattere e come possono essere modificati nelle partiture.

## Tipi di carattere in dettaglio

I tipi di carattere vengono gestiti attraverso varie librerie. FontConfig rileva i tipi di carattere disponibili nel sistema; i tipi selezionati sono riprodotti con Pango.

I tipi di carattere della notazione musicale possono essere descritti come un insieme di glifi specifici, ordinati in varie famiglie. La seguente sintassi permette di usare vari caratteri **feta** di LilyPond (non testuali) direttamente nella modalità markup:

```
a'1^\markup {
  \vcenter {
    \override #'(font-encoding . fetaBraces)
    \lookup #"brace120"
    \override #'(font-encoding . fetaText)
    \column { 1 3 sf }
    \override #'(font-encoding . fetaMusic)
    \lookup #"noteheads.s0petrucci"
  }
}
```



Tuttavia, tutti questi glifi, ad eccezione delle graffe di varie dimensioni contenute in **fetaBraces**, sono già utilizzabili con la sintassi ben più semplice descritta in `<undefined>` [Music notation inside markup], pagina `<undefined>`.

Quando si usano i glifi contenuti in **fetaBraces**, la dimensione della graffa viene specificata dalla parte numerica del nome del glifo, in unità arbitrarie. Può essere specificato qualsiasi numero intero da 0 a 575 compresi, dove 0 corrisponde alla graffa più piccola. Il valore ottimale

deve essere determinato per tentativi. Questi glifi sono tutte graffe sinistre; le graffe destre si possono ottenere con la rotazione, vedi Sezione 5.4.9 [Rotating objects], pagina 625.

Sono disponibili tre famiglie di tipi di carattere:

- Il tipo *roman* (con grazie), il cui valore predefinito è LilyPond Serif (un alias di TeX Gyre Schola).
- Il tipo *sans* (senza grazie), il cui valore predefinito è LilyPond Sans Serif (un alias di TeX Gyre Heros).
- Il tipo monospaziato *typewriter*, il cui valore predefinito è LilyPond Monospace (un alias di TeX Gyre Cursor).

Ogni famiglia può avere forme e serie differenti. L'esempio seguente illustra la possibilità di scegliere famiglie, forme, serie e dimensioni alternative. Il valore specificato per **font-size** è la modifica relativa alla dimensione predefinita.

```
\override Score.RehearsalMark.font-family = #'typewriter
\mark \markup "Ouverture"
\override Voice.TextScript.font-shape = #'italic
\override Voice.TextScript.font-series = #'bold
d''2.^{\markup "Allegro"}
\override Voice.TextScript.font-size = #-3
c''4^smaller
```



Una sintassi simile si usa nella modalità markup; tuttavia in questo caso è preferibile usare la sintassi più semplice spiegata in <undefined> [Selecting font and font size], pagina <undefined>:

```
\markup {
  \column {
    \line {
      \override #'(font-shape . italic)
      \override #'(font-size . 4)
      Idomeneo,
    }
    \line {
      \override #'(font-family . typewriter)
      {
        \override #'(font-series . bold)
        re
        di
      }
      \override #'(font-family . sans)
      Creta
    }
  }
}
```

*Idomeneo,*  
re di Creta

Sebbene sia semplice passare a un tipo di carattere preconfigurato, è anche possibile usare altri tipi, come viene spiegato nelle sezioni successive: [\[Single entry fonts\]](#), pagina [\[undefined\]](#), e [\[Entire document fonts\]](#), pagina [\[undefined\]](#).

## Vedi anche

Guida alla notazione: [\[The Feta font\]](#), pagina [\[undefined\]](#), [\[Music notation inside markup\]](#), pagina [\[undefined\]](#), Sezione 5.4.9 [\[Rotating objects\]](#), pagina 625, [\[Selecting font and font size\]](#), pagina [\[undefined\]](#), Sezione A.11.1 [\[Font\]](#), pagina 683.

## Tipi di carattere per singolo oggetto

Si può usare nella partitura qualsiasi tipo di carattere che sia installato nel sistema operativo e riconosciuto da FontConfig, usando la seguente sintassi:

```
\override Staff.TimeSignature.font-name = #"Bitstream Charter"
\override Staff.TimeSignature.font-size = #2
\time 3/4

a'1_\markup {
  \override #'(font-name . "Bitstream Vera Sans,sans-serif, Oblique Bold")
    { Vera Oblique Bold }
}
```



*font-name* può essere definito da una lista separata da virgola di ‘font’ e una lista separata da spazi di ‘stili’. Se il ‘font’ nella lista è installato e contiene il glifo richiesto, verrà usato, altrimenti sarà usato al suo posto il font *successivo*.

Lanciando lilypond con la seguente opzione si ottiene un elenco di tutti i tipi di carattere disponibili nel sistema operativo:

```
lilypond -dshow-available-fonts x
```

## Vedi anche

Guida alla notazione: [\[Fonts explained\]](#), pagina [\[undefined\]](#), [\[Entire document fonts\]](#), pagina [\[undefined\]](#).

Frammenti: Sezione “Text” in *Frammenti di codice*.

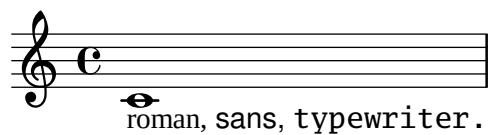
## Tipi di carattere per l’intero documento

È possibile modificare i tipi di carattere usati come tipi predefiniti nelle famiglie *roman*, *sans* e *typewriter* specificandoli, in questo ordine, come è mostrato nell’esempio seguente, che ridimensiona automaticamente i caratteri col valore impostato per la dimensione globale del rigo. In modo analogo a [\[Single entry fonts\]](#), pagina [\[undefined\]](#), si può indicare con una lista separata da virgole di ‘font’. Gli ‘stili’ dei font, invece, non possono essere definiti. I tipi di carattere sono spiegati in [\[Fonts explained\]](#), pagina [\[undefined\]](#).

```
\paper {
  #(define fonts
    (make-pango-font-tree "Times New Roman"
      "Nimbus Sans,Nimbus Sans L"
      "Luxi Mono"
      (/ staff-height pt 20)))
```

}

```
\relative c'{
  c1-\markup {
    roman,
    \sans sans,
    \typewriter typewriter. }
}
```



## Vedi anche

Guida alla notazione: [\[Fonts explained\]](#), pagina [\[undefined\]](#), [\[Single entry fonts\]](#), pagina [\[undefined\]](#), [\[Selecting font and font size\]](#), pagina [\[undefined\]](#), Sezione A.11.1 [\[Font\]](#), pagina 683.

## 2 Notazione specialistica

Questo capitolo spiega come creare la notazione musicale per particolari tipi di strumento o per stili specifici.

### 2.1 Musica vocale

**Recitativo**  
Baritono

216



O Freun - - de, nicht die - se Töne!

222



Sondern laßt uns an - - ge -

228



nehmere an - stimmen, und freu -

232



- - - - - denvollere!

Questa sezione spiega come scrivere la musica vocale e assicurarsi che il testo sia allineato con le note della melodia.

#### 2.1.1 Notazione comune per la musica vocale

Questa sezione tratta le questioni relative alla maggior parte delle tipologie di musica vocale.

#### Riferimenti per la musica vocale

Questa sezione indica dove trovare informazioni dettagliate sulle questioni comuni a qualsiasi tipo di musica vocale.

- La maggior parte degli stili di musica vocale usa il testo semplice per le parti vocali. Un'introduzione a questo tipo di notazione è disponibile in Sezione “Impostare canzoni semplici” in *Manuale di Apprendimento*.
- La musica vocale richiede solitamente l'uso della modalità **markup**, sia per il testo musicale che per altri elementi testuali (i nomi dei personaggi, etc.). Questa sintassi è spiegata in [\[Text markup introduction\]](#), pagina [\[Text markup introduction\]](#).
- L'*Ambitus* può essere aggiunto all'inizio dei righe per la voce, come è spiegato in [\[Ambitus\]](#), pagina 35.
- Le indicazioni dinamiche sono di norma posizionate sotto il rigo, ma nella musica corale sono posizionate solitamente sopra il rigo per evitare il testo, come è spiegato in [\[Score layouts for choral\]](#), pagina [\[Score layouts for choral\]](#).

## Vedi anche

Glossario musicale: Sezione “ambitus” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Impostare canzoni semplici” in *Manuale di Apprendimento*.

Guida alla notazione: [\[Text markup introduction\]](#), pagina [\[Ambitus\]](#), pagina 35, [\[Score layouts for choral\]](#), pagina [\[Score layouts for choral\]](#).

Frammenti: Sezione “Vocal music” in *Frammenti di codice*.

## Inserimento del testo vocale

Il testo vocale viene inserito in una speciale modalità di input, che può essere introdotta dai comandi `\lyricmode`, `\addlyrics` o `\lyricsto`. In questa speciale modalità, l'input `d` non viene analizzato come l'altezza *Re*, ma come una sillaba di una lettera. In altre parole, le sillabe si inseriscono come le note, ma le altezze sono sostituite dal testo.

Per esempio:

```
\lyricmode { Three4 blind mice,2 three4 blind mice2 }
```

Ci sono due metodi principali per determinare il posizionamento orizzontale delle sillabe. Si può indicare la durata di ogni sillaba esplicitamente, come nell'esempio precedente; oppure si può lasciare che il testo sia allineato automaticamente a una melodia o a un'altra voce del brano, usando `\addlyrics` o `\lyricsto`. Il primo metodo è descritto sotto in [\[Manual syllable durations\]](#), pagina [\[Manual syllable durations\]](#); il secondo in [\[Automatic syllable durations\]](#), pagina [\[Automatic syllable durations\]](#).

Una parola o sillaba del testo inizia con un carattere alfabetico (seguito da altri caratteri, vedi dopo) ed è terminata da uno spazio bianco o da un numero. I caratteri successivi al primo nella sillaba possono essere un qualsiasi carattere che non sia un numero o uno spazio bianco.

Dato che qualsiasi carattere che non sia un numero o uno spazio viene considerato come parte della sillaba, una parola è valida anche se finisce con `}`, causando spesso il seguente errore:

```
\lyricmode { lah lah lah}
```

In questo esempio, la parentesi `}` è inclusa nella sillaba finale, dunque la parentesi iniziale non viene chiusa e la compilazione del file probabilmente non riuscirà. Le parentesi devono quindi essere sempre circondate da uno spazio:

```
\lyricmode { lah lah lah }
```

La punteggiatura, i caratteri accentati, quelli di lingue diverse dall'inglese e i caratteri speciali (come il simbolo del cuore o le virgolette oblique) possono essere inseriti direttamente nel file di input, purché sia salvato nella codifica UTF-8. Ulteriori informazioni in [\[Special characters\]](#), pagina [\[Special characters\]](#).

```
\relative { d''8 c16 a bes8 f e' d c4 }
\addlyrics { „Schad' um das schön -- ne grü -- ne Band, }
```



Le virgolette normali possono essere usate nel testo vocale, ma devono essere precedute da un carattere di barra inversa e l'intera sillaba deve essere racchiusa tra altre virgolette. Per esempio,

```
\relative { \time 3/4 e'4 e4. e8 d4 e d c2. }
\addlyrics { "\"I" am so lone -- "ly,\" said she }
```



La definizione completa di come può iniziare una parola nella modalità testo è un po' più complessa. Può iniziare con un carattere alfabetico, `_`, `?`, `!`, `:`, `'`, i caratteri di controllo `^A` fino a `^F`, `^Q` fino a `^W`, `^Y`, `^^`, qualsiasi carattere a 8-bit con un codice ASCII superiore a 127 oppure una combinazione a due caratteri di una barra inversa seguita da ```, `'`, `"` o `^`.

Per avere un maggior controllo sull'aspetto del testo si può usare `\markup` all'interno del testo. Per una spiegazione delle molte opzioni disponibili leggere `<undefined>` [Formatting text], pagina `<undefined>`.

## Frammenti di codice selezionati

### *Formattazione delle sillabe del testo vocale*

La modalità markup può essere usata per formattare le singole sillabe del testo vocale.

```
mel = \relative c'' { c4 c c c }
lyr = \lyricmode {
  Lyrics \markup { \italic can } \markup { \with-color #red contain }
  \markup { \fontsize #8 \bold Markup! }
}

<<
  \new Voice = melody \mel
  \new Lyrics \lyricsto melody \lyr
>>
```



## Vedi anche

Manuale di apprendimento: Sezione “Canzoni” in *Manuale di Apprendimento*.

Guida alla notazione: `<undefined>` [Automatic syllable durations], pagina `<undefined>`, `<undefined>` [Fonts], pagina `<undefined>`, `<undefined>` [Formatting text], pagina `<undefined>`, Sezione 5.4.1 [Input modes], pagina 610, `<undefined>` [Manual syllable durations], pagina `<undefined>`, `<undefined>` [Special characters], pagina `<undefined>`.

Guida al funzionamento interno: Sezione “LyricText” in *Guida al Funzionamento Interno*.

Frammenti: Sezione “Text” in *Frammenti di codice*.

## Allineamento del testo alla melodia

Il testo vocale viene interpretato in `\lyricmode` e stampato in un contesto `Lyrics`, vedi Sezione 5.1.1 [Contexts explained], pagina 579.

```
\new Lyrics \lyricmode { ... }
```

Due varianti di `\lyricmode` impostano un contesto associato usato per sincronizzare le sillabe del testo con la musica. Il più comodo `\addlyrics` segue immediatamente il contenuto musicale del contesto della voce con cui deve essere sincronizzato, creando implicitamente un contesto `Lyrics`. Il più versatile `\lyricsto` richiede sia di specificare il contesto della voce associata con un nome sia di creare esplicitamente un contesto `Lyrics` che contenga il testo. Maggiori dettagli in `<undefined>` [Automatic syllable durations], pagina `<undefined>`.



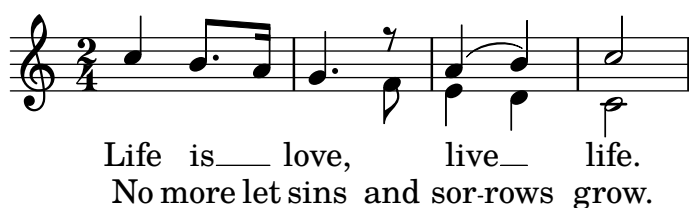
Ci sono due modi per allineare il testo a una melodia:

- Il testo può essere allineato automaticamente in modo che le durate delle sillabe siano prese da un'altra voce o (in circostanze speciali) da una melodia associata, usando `\addlyrics`, `\lyricsto` o impostando la proprietà `associatedVoice`. Ulteriori informazioni in [\[Automatic syllable durations\]](#), pagina [\[Automatic syllable durations\]](#).

```
<<
\new Staff <<
  \time 2/4
  \new Voice = "one" \relative {
    \voiceOne
    c''4 b8. a16 g4. r8 a4 ( b ) c2
  }
  \new Voice = "two" \relative {
    \voiceTwo
    s2 s4. f'8 e4 d c2
  }
>>

% prende le durate e l'allineamento dalle note in "one"
\new Lyrics \lyricsto "one" {
  Life is __ _ love, live __ life.
}

% prende le durate e l'allineamento dalle note in "one" inizialmente
% poi passa a "two"
\new Lyrics \lyricsto "one" {
  No more let
  \set associatedVoice = "two" % deve essere impostato prima della sillaba
  sins and sor -- rows grow.
}
>>
```



La prima strofa mostra il modo normale di inserire il testo.

La seconda strofa mostra come cambiare la voce da cui prendere le durate per le sillabe del testo. Ciò è utile se le parole di strofe diverse si distribuiscono lungo le note in modo differente e tutte le durate sono disponibili nei contesti della voce. Ulteriori dettagli in [\[Stanzas\]](#), pagina [\[Stanzas\]](#).

- Il testo può essere allineato indipendentemente dalla durata delle note se le durate delle sillabe vengono specificate esplicitamente e inserite con `\lyricmode`.

```
<<
\new Voice = "one" \relative {
  \time 2/4
  c''4 b8. a16 g4. f8 e4 d c2
}
>>
```

```
% usa la durata esplicita precedente di 2;
\new Lyrics \lyricmode {
  Joy to the earth!
}

% durate esplicite, impostate su un ritmo diverso
\new Lyrics \lyricmode {
  Life4 is love,2. live4 life.2
}
>>
```



La prima strofa non è allineata con le note perché le durate non sono state specificate, e il valore precedente di 2 viene usato per ogni parola.

La seconda strofa mostra come le parole possano essere allineate in modo del tutto indipendente dalle note. Ciò è utile se le parole di strofe diverse si distribuiscono lungo le note in modo differente e le durate necessarie non sono disponibili in un contesto musicale. Ulteriori dettagli in [\[Manual syllable durations\]](#), pagina [\[undefined\]](#). Questa tecnica è utile anche quando si imposta un dialogo sopra una musica, come si può vedere negli esempi in [\[Dialogue over music\]](#), pagina [\[undefined\]](#).

## Vedi anche

Manuale di apprendimento: Sezione “Allineare il testo alla melodia” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.1.1 [\[Contexts explained\]](#), pagina 579, [\[Automatic syllable durations\]](#), pagina [\[undefined\]](#), [\[Stanzas\]](#), pagina [\[undefined\]](#), [\[Manual syllable durations\]](#), pagina [\[undefined\]](#), [\[Dialogue over music\]](#), pagina [\[undefined\]](#), [\[Manual syllable durations\]](#), pagina [\[undefined\]](#).

Guida al funzionamento interno: Sezione “Lyrics” in *Guida al Funzionamento Interno*.

## Durate automatiche delle sillabe

Il testo vocale può essere allineato automaticamente alle note di una melodia in tre modi:

- specificando il nome del contesto Voice contenente la melodia con `\lyricsto`,
- introducendo il testo con `\addlyrics` e inserendolo subito dopo il contesto Voice contenente la melodia,
- impostando la proprietà `associatedVoice`, l’allineamento del testo può essere cambiato su un contesto Voice con un nome diverso in qualsiasi momento musicale.

In tutti questi tre metodi è possibile disegnare dei trattini tra le sillabe di una parola e delle linee di estensione oltre la fine di una parola. Maggiori dettagli in [\[Extenders and hyphens\]](#), pagina [\[undefined\]](#).

Il contesto Voice contenente la melodia al quale il testo si deve allineare non deve essere “morto”, altrimenti il testo successivo a quel punto verrà perduto. Questo può accadere se ci sono dei momenti in cui quella voce non ha nulla da fare. Metodi per mantenere attivi i contesti sono descritti in Sezione 5.1.3 [\[Keeping contexts alive\]](#), pagina 584.

## Uso di `\lyricsto`

Il testo vocale può essere allineato a una melodia automaticamente specificando il nome del contesto voce con `\lyricsto`:

```
<<
  \new Voice = "melody" \relative {
    a'1 a4. a8 a2
  }
  \new Lyrics \lyricsto "melody" {
    These are the words
  }
>>
```



In questo modo il testo viene allineato alle note del contesto `Voice` con quel nome, che deve già esistere. Ecco perché di solito il contesto `Voice` viene indicato prima, seguito dal contesto `Lyrics`. Il testo segue il comando `\lyricsto`. Il comando `\lyricsto` invoca la modalità testo automaticamente. Per impostazione predefinita, il testo viene posizionato sotto le note. Altri posizionamenti sono descritti in [\[Placing lyrics vertically\]](#), pagina [\[undefined\]](#).

## Uso di `\addlyrics`

Il comando `\addlyrics` è solo una comoda scorciatoia da usare per evitare di impostare il testo tramite una struttura più complessa.

```
{ MUSICA }
\addlyrics { TESTO VOCALE }
```

equivale a

```
\new Voice = "blah" { MUSICA }
\new Lyrics \lyricsto "blah" { TESTO VOCALE }
```

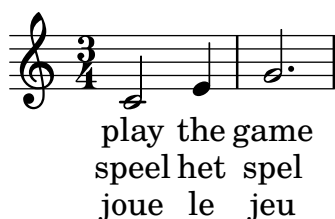
```
Ecco un esempio,
{
  \time 3/4
  \relative { c'2 e4 g2. }
  \addlyrics { play the game }
}
```



Si possono aggiungere più strofe aggiungendo più sezioni `\addlyrics`:

```
{
  \time 3/4
  \relative { c'2 e4 g2. }
  \addlyrics { play the game }
  \addlyrics { speel het spel }
  \addlyrics { joue le jeu }
```

}



Il comando `\addlyrics` non può gestire le impostazioni polifoniche. Inoltre non può essere usato per associare il testo alla voce per intavolatura (`TabVoice`). In questi casi bisogna usare `\lyricsto`.

## Uso di `associatedVoice`

La melodia a cui allineare il testo vocale può essere cambiata impostando la proprietà `associatedVoice`,

```
\set associatedVoice = #"lala"
```

Il valore della proprietà (qui: `"lala"`) deve essere il nome di un contesto `Voice`. Per ragioni tecniche, il comando `\set` deve essere posizionato una sillaba prima prima di quella alla quale si riferisce il cambio di voce.

Ecco un esempio che ne dimostra l'utilizzo:

```
<<
\new Staff <<
  \time 2/4
  \new Voice = "one" \relative {
    \voiceOne
    c''4 b8. a16 g4. r8 a4 ( b ) c2
  }
  \new Voice = "two" \relative {
    \voiceTwo
    s2 s4. f'8 e8 d4. c2
  }
  }
>>
% inizialmente prende le note e le durate da "one"
% poi passa a "two"
\new Lyrics \lyricsto "one" {
  No more let
  \set associatedVoice = "two" % must be set one syllable early
  sins and sor -- rows grow.
}
>>
```



## Vedi anche

Guida alla notazione: [\[Extenders and hyphens\]](#), pagina [\[undefined\]](#), Sezione 5.1.3 [\[Keeping contexts alive\]](#), pagina 584, [\[undefined\]](#) [\[Placing lyrics vertically\]](#), pagina [\[undefined\]](#).

## Durate manuali delle sillabe

In alcune musiche vocali complesse, si potrebbe voler posizionare il testo in modo totalmente indipendente dalle note. In tali casi non bisogna usare `\lyricsto` o `\addlyrics` e nemmeno impostare `associatedVoice`. Le sillabe verranno invece inserite come se fossero note, ma col testo al posto delle altezze, e la durata di ogni sillaba sarà indicata esplicitamente dopo la sillaba.

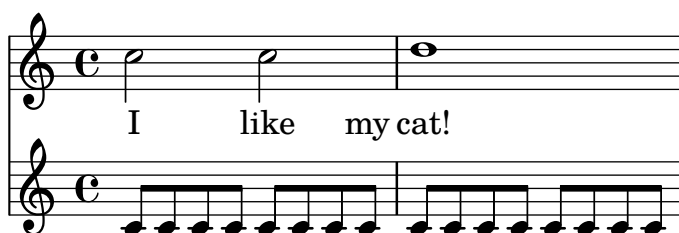
Si possono mostrare le linee tratteggiate tra le sillabe come sempre, mentre le linee di estensione non appaiono se non c'è una voce associata.

Ecco due esempi:

```
<<
\new Voice = "melody" \relative {
  c''2 a f f e e
}
\new Lyrics \lyricmode {
  c4. -- a -- f -- f -- e2. -- e
}
>>
```



```
<<
\new Staff {
  \relative {
    c''2 c2
    d1
  }
}
\new Lyrics {
  \lyricmode {
    I2 like4. my8 cat!1
  }
}
\new Staff {
  \relative {
    c'8 c c c c c c c
    c8 c c c c c c c
  }
}
>>
```



Questa tecnica è utile quando si scrivono dialoghi , vedi `\dialogue` [Dialogue over music], pagina `\dialogue`.

Per cambiare l'allineamento delle sillabe, basta impostare la proprietà `self-alignment-X`:

```
<<
\new Voice = "melody" \relative {
  \time 3/4
  c'2 e4 g2 f
}
\new Lyrics \lyricmode {
  \override LyricText.self-alignment-X = #LEFT
  play1 a4 game4
}
>>
```



## Vedi anche

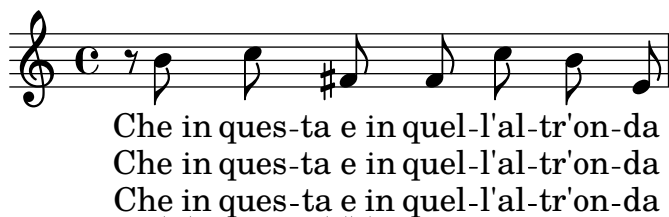
Guida alla notazione: [\[Dialogue over music\]](#), pagina [\[undefined\]](#).

Guida al funzionamento interno: Sezione “Lyrics” in *Guida al Funzionamento Interno*, Sezione “Voice” in *Guida al Funzionamento Interno*.

## Più sillabe in una nota

Per assegnare più di una sillaba a una singola nota mantenendo uno spazio tra le sillabe, occorre mettere la frase tra virgolette o usare il carattere `_`. Altrimenti si può usare il simbolo tilde (`~`) per ottenere una legatura di valore per il testo.

```
{
  \relative {
    \autoBeamOff
    r8 b' c fis, fis c' b e,
  }
  \addlyrics
  {
    \override LyricHyphen.minimum-distance = #1.0 % Ensure hyphens are visible
    Che_in ques -- ta_e_in quel -- l'al -- tr'on -- da
  }
  \addlyrics { "Che in" ques -- "ta e in" quel -- l'al -- tr'on -- da }
  \addlyrics { Che~in ques -- ta~e~in quel -- l'al -- tr'on -- da }
}
```



## Vedi anche

Guida al funzionamento interno: Sezione “LyricCombineMusic” in *Guida al Funzionamento Interno*.

## Più note in una sillaba

Talvolta, in particolare nella musica medievale e barocca, molte note vengono cantate in una sillaba; si chiama melisma (vedi Sezione “melisma” in *Glossario Musicale*). La sillaba di un melisma viene solitamente allineata a sinistra della prima nota del melisma.

Quando un melisma si trova su una sillaba diversa dall’ultima in una parola, quella sillaba di solito viene collegata a quella successiva con una linea tratteggiata. Ciò si indica inserendo un doppio trattino --, subito dopo la sillaba.

Altrimenti, se un melisma si trova sull’ultima o unica sillaba in una parola, solitamente appare una linea di estensione dalla fine della sillaba all’ultima nota del melisma. Ciò si indica inserendo un doppio trattino basso, \_\_, subito dopo la parola.

Esistono cinque modi per indicare i melismi:

- I melismi vengono creati automaticamente sulle note legate insieme:

```
<<
\new Voice = "melody" \relative {
  \time 3/4
  f''4 g2 ~ |
  4 e2 ~ |
  8
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- e __
}
>>
```



- I melismi possono essere creati automaticamente dalla musica inserendo delle legature di portamento sulle note di ogni melisma. Questo è il modo più comune di inserire il testo:

```
<<
\new Voice = "melody" \relative {
  \time 3/4
  f''4 g8 ( f e f )
  e8 ( d e2 )
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- e __
}
>>
```



Attenzione: le legature di frase non causano la creazione di melismi.

- Le note sono considerate un melisma se le loro travature sono disposte manualmente, purché la travatura automatica sia disabilitata. Vedi [\[Setting automatic beam behavior\]](#), pagina [\[Setting automatic beam behavior\]](#).

```
<<
```

```

\new Voice = "melody" \relative {
  \time 3/4
  \autoBeamOff
  f''4 g8[ f e f]
  e2.
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- e
}
>>

```



Ovviamente ciò non è adatto ai melismi su note più lunghe degli ottavi.

- Un gruppo di note privo di legature sarà trattato come un melisma se sono comprese tra `\melisma` `\melismaEnd`.

```

<<
\new Voice = "melody" \relative {
  \time 3/4
  f''4 g8
  \melisma
  f e f
  \melismaEnd
  e2.
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- e
}
>>

```



- Un melisma può essere definito interamente nel testo inserendo un solo trattino basso, `_`, per ogni nota ulteriore da aggiungere al melisma.

```

<<
\new Voice = "melody" \relative {
  \time 3/4
  f''4 g8 f e f
  e8 d e2
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- _ _ _ e _ _ _
}
>>

```





È possibile avere legature di portamento e di valore e travature manuali nella melodia senza che indichino i melismi. Per farlo si imposta `melismaBusyProperties`:

```
<<
\new Voice = "melody" \relative {
  \time 3/4
  \set melismaBusyProperties = #'()
  c'4 d ( e )
  g8 [ f ] f4 ~ 4
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- e e -- le -- i -- son
}
>>
```



Altre impostazioni di `melismaBusyProperties` possono essere usate per includere o escludere selettivamente legature di valore e di portamento e travature dal rilevamento automatico del melisma; consultare `melismaBusyProperties` in Sezione “Tunable context properties” in *Guida al Funzionamento Interno*.

Altrimenti, se si vuole ignorare tutte le indicazioni di melisma, basta impostare `ignoreMelismata` su vero; vedi `\undefined` [Stanzas with different rhythms], pagina `\undefined`.

Se un melisma è necessario in un passaggio in cui `melismaBusyProperties` è attivo, lo si può indicare inserendo un singolo trattino basso nel testo per ogni nota che debba essere inclusa nel melisma:

```
<<
\new Voice = "melody" \relative {
  \time 3/4
  \set melismaBusyProperties = #'()
  c'4 d ( e )
  g8 [ f ] ~ f4 ~ 4
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- _ e _ _ _ _
}
>>
```



## Comandi predefiniti

`\autoBeamOff`, `\autoBeamOn`, `\melisma`, `\melismaEnd`.

## Vedi anche

Glossario musicale: Sezione “melisma” in *Glossario Musicale*.

Manuale di apprendimento: Sezione “Allineare il testo alla melodia” in *Manuale di Apprendimento*.

Guida alla notazione: `<undefined>` [Aligning lyrics to a melody], pagina `<undefined>`, `<undefined>` [Automatic syllable durations], pagina `<undefined>`, `<undefined>` [Setting automatic beam behavior], pagina `<undefined>`, `<undefined>` [Stanzas with different rhythms], pagina `<undefined>`.

Guida al funzionamento interno: Sezione “Tunable context properties” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Le linee di estensione sotto i melismi non vengono creati automaticamente; devono essere inseriti a mano con un doppio trattino basso.

## Estensori e trattini

Nell’ultima sillaba di una parola, i melismi sono talvolta indicati con una lunga linea orizzontale che inizia dalla sillaba del melisma e termina in quella successiva. Tale linea viene chiamata linea di estensione e si inserisce con ‘`--`’ (notare gli spazi prima e dopo i due caratteri).

**Nota:** I melismi appaiono nella partitura come linee di estensione, che si inseriscono con un doppio trattino basso; ma i melismi brevi si possono inserire anche saltando note singole but short melismata can also be entered by skipping individual notes, which are entered as single underscore characters; these do not make an extender line to be typeset by default.

Il trattino centrato si inserisce con ‘`--`’ tra le sillabe di una stessa parola (notare gli spazi prima e dopo i due caratteri). Il trattino sarà centrato tra le sillabe e la sua lunghezza sarà regolata a seconda dello spazio tra le sillabe.

Nelle partiture compresse i trattini possono essere eliminati. Questo comportamento è controllato da due proprietà di **LyricHyphen**: `minimum-distance` (distanza minima tra le due sillabe) e `minimum-length` (soglia sotto la quale i trattini vengono rimossi).

## Vedi anche

Guida al funzionamento interno: Sezione “LyricExtender” in *Guida al Funzionamento Interno*, Sezione “LyricHyphen” in *Guida al Funzionamento Interno*.

### 2.1.2 Tecniche specifiche per il testo vocale

#### Lavorare con testo e variabili

Si possono creare delle variabili contenenti il testo vocale, ma questo deve essere inserito in modalità testo vocale:

```
musicOne = \relative {
  c' '4 b8. a16 g4. f8 e4 d c2
}
verseOne = \lyricmode {
  Joy to the world, the Lord is come.
}
\score {
  <<
  \new Voice = "one" {
    \time 2/4
```

```

    \musicOne
  }
  \new Lyrics \lyricsto "one" {
    \verseOne
  }
  >>
}

```



Non è necessario aggiungere le durate se la variabile viene richiamata con `\addlyrics` o `\lyricsto`. Se la musica ha un ordine diverso e più complesso, conviene definire prima le variabili che contengono la musica e il testo, poi impostare la gerarchia di righe e testo, omettendo il testo stesso, e infine aggiungere il testo all'interno di un blocco `\context`. Ciò garantisce che le voci richiamate da `\lyricsto` siano sempre state definite prima. Per esempio:

```

sopranoMusic = \relative { c''4 c c c }
contraltoMusic = \relative { a'4 a a a }
sopranoWords = \lyricmode { Sop -- ra -- no words }
contraltoWords = \lyricmode { Con -- tral -- to words }

```

```

\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice = "sopranos" {
        \sopranoMusic
      }
    }
    \new Lyrics = "sopranos"
    \new Lyrics = "contraltos"
    \new Staff {
      \new Voice = "contraltos" {
        \contraltoMusic
      }
    }
    \context Lyrics = "sopranos" {
      \lyricsto "sopranos" {
        \sopranoWords
      }
    }
    \context Lyrics = "contraltos" {
      \lyricsto "contraltos" {
        \contraltoWords
      }
    }
  }
  >>
}

```



## Vedi anche

Guida alla notazione: [\[Placing lyrics vertically\]](#), pagina [\[undefined\]](#).

Guida al funzionamento interno: Sezione “LyricCombineMusic” in *Guida al Funzionamento Interno*, Sezione “Lyrics” in *Guida al Funzionamento Interno*.

## Posizionamento verticale del testo

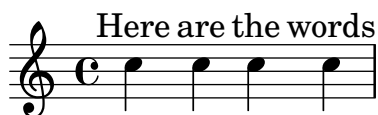
A seconda del tipo di musica, il testo può trovarsi sopra o sotto il rigo oppure tra i righi. Posizionare il testo sotto il rigo associato è il modo più semplice, infatti basta definire il contesto Lyrics sotto il contesto Staff;

```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



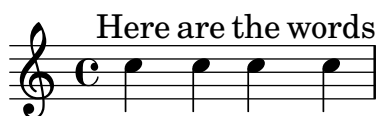
Per posizionare il testo sopra il rigo ci sono due metodi. Il metodo più semplice (e preferito) consiste nell’usare la stessa sintassi precedente e indicare in modo esplicito la posizione del testo:

```
\score {
  <<
    \new Staff = "staff" {
      \new Voice = "melody" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics \with { alignAboveContext = "staff" } {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



Altrimenti si possono usare due passaggi. Prima si dichiara il contesto Lyrics (privo di contenuto) prima dei contesti Staff e Voice, poi il comando `\lyricsto` viene posizionato dopo la voce a cui si riferisce tramite `\context`. Ecco un esempio:

```
\score {
  <<
    \new Lyrics = "lyrics" \with {
      % il testo sopra un rigo deve avere questo override
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
    \new Staff {
      \new Voice = "melody" {
        \relative { c''4 c c c }
      }
    }
    \context Lyrics = "lyrics" {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



Quando ci sono due voci in righe separate, si può posizionare il testo tra i righe usando uno di questi metodi. Ecco un esempio del secondo metodo:

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice = "sopranos" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics = "sopranos"
    \new Lyrics = "contraltos" \with {
      % il testo sopra un rigo deve avere questo override
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
    \new Staff {
      \new Voice = "contraltos" {
        \relative { a'4 a a a }
      }
    }
  >>
  \context Lyrics = "sopranos" {
    \lyricsto "sopranos" {
      Sop -- ra -- no words
    }
  }
}
```

```

\context Lyrics = "contraltos" {
  \lyricsto "contraltos" {
    Con -- tral -- to words
  }
}
>>
}

```



Si possono generare altre combinazioni di testo e righe a partire da questi esempi oppure studiando i modelli nel Manuale di apprendimento, vedi Sezione “Modelli per gruppi vocali” in *Manuale di Apprendimento*.

## Frammenti di codice selezionati

*Ottenere la spaziatura del testo della vecchia versione 2.12*

Il motore di spaziatura verticale è cambiato a partire dalla versione 2.14. Ciò può far sì che il testo vocale abbia un posizionamento diverso.

È possibile impostare delle proprietà dei contesti **Lyric** e **Staff** che facciano sì che il motore di spaziatura si comporti come nella versione 2.12.

```

global = {
  \key d \major
  \time 3/4
}

sopMusic = \relative c' {
  % VERSE ONE
  fis4 fis fis | \break
  fis4. e8 e4
}

altoMusic = \relative c' {
  % VERSE ONE
  d4 d d |
  d4. b8 b4 |
}

tenorMusic = \relative c' {
  a4 a a |
  b4. g8 g4 |
}

bassMusic = \relative c {
  d4 d d |
  g,4. g8 g4 |
}

```

```

}

words = \lyricmode {
  Great is Thy faith- ful- ness,
}

\score {
  \new ChoirStaff <<
    \new Lyrics = sopranos
    \new Staff = women <<
      \new Voice = "sopranos" {
        \voiceOne
        \global \sopMusic
      }
      \new Voice = "altos" {
        \voiceTwo
        \global \altoMusic
      }
    >>
    \new Lyrics = "altos"
    \new Lyrics = "tenors"
    \new Staff = men <<
      \clef bass
      \new Voice = "tenors" {
        \voiceOne
        \global \tenorMusic
      }
      \new Voice = "basses" {
        \voiceTwo \global \bassMusic
      }
    >>
    \new Lyrics = basses
    \context Lyrics = sopranos \lyricsto sopranos \words
    \context Lyrics = altos \lyricsto altos \words
    \context Lyrics = tenors \lyricsto tenors \words
    \context Lyrics = basses \lyricsto basses \words
  >>
  \layout {
    \context {
      \Lyrics
      \override VerticalAxisGroup.staff-affinity = ##f
      \override VerticalAxisGroup.staff-staff-spacing =
        #'((basic-distance . 0)
          (minimum-distance . 2)
          (padding . 2))
    }
    \context {
      \Staff
      \override VerticalAxisGroup.staff-staff-spacing =
        #'((basic-distance . 0)
          (minimum-distance . 2)
          (padding . 2))
    }
  }
}

```

}  
}  
}

Great is Thy

Great is Thy

Great is Thy

faith- ful- ness,

faith- ful- ness,

faith- ful- ness,

faith- ful- ness,

## Vedi anche

Manuale di apprendimento: Sezione “Modelli per gruppi vocali” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.1.7 [Context layout order], pagina 596, Sezione 5.1.2 [Creating and referencing contexts], pagina 581.

## Posizionamento orizzontale delle sillabe

Per aumentare lo spazio tra le righe del testo, si imposta la proprietà `minimum-distance` di `LyricSpace`.

```
\relative c' {
  c c c c
  \override Lyrics.LyricSpace.minimum-distance = #1.0
  c c c c
}
\addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
}
```



}



Per applicare questa modifica a tutti i testi della partitura, impostare la proprietà nel blocco `\layout`.

```
\score {
  \relative {
    c' c c c
    c c c c
  }
  \addlyrics {
    longtext longtext longtext longtext
    longtext longtext longtext longtext
  }
  \layout {
    \context {
      \Lyrics
      \override LyricSpace.minimum-distance = #1.0
    }
  }
}
```



## Frammenti di codice selezionati

### *Allineamento del testo vocale*

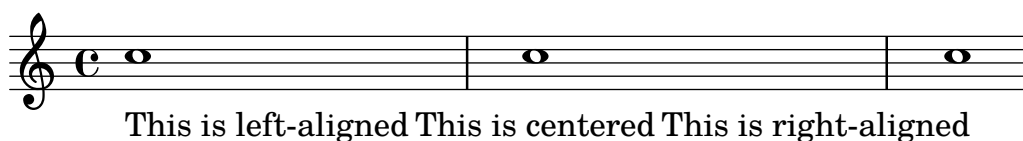
L'allineamento orizzontale del testo vocale si imposta attraverso la proprietà `self-alignment-X` dell'oggetto `LyricText`. `#-1` è sinistra, `#0` è centro e `#1` è destra; si possono usare anche `#LEFT`, `#CENTER` e `#RIGHT`.

```
\layout { ragged-right = ##f }
\relative c' {
  c1
```

```

c1
c1
}
\addlyrics {
  \once \override LyricText.self-alignment-X = #LEFT
  "This is left-aligned"
  \once \override LyricText.self-alignment-X = #CENTER
  "This is centered"
  \once \override LyricText.self-alignment-X = #1
  "This is right-aligned"
}

```



Verificare che le annotazioni testuali e il testo vocale stiano dentro i margini richiede ulteriori calcoli. Se si desidera velocizzare un po' l'elaborazione, tale funzionalità può essere disabilitata:

```
\override Score.PaperColumn.keep-inside-line = ##f
```

Per far sì che il testo eviti anche le stanghette, usare

```

\layout {
  \context {
    \Lyrics
    \consists "Bar_engraver"
    \consists "Separating_line_group_engraver"
    \hide BarLine
  }
}

```

## Testo e ripetizioni

### Ripetizioni semplici

Le ripetizioni in *musica* sono trattate in un'altra sezione: [\[Repeats\]](#), pagina [\[Repeats\]](#). Questa sezione spiega come aggiungere del testo vocale a delle parti musicali ripetute.

Il testo che si riferisce a una sezione musicale ripetuta deve avere lo stesso costruito di ripetizione della musica, se le parole non sono modificate.

```

\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat volta 2 { b4 b b b }
        }
      }
    }
  }
  \new Lyrics {
    \lyricsto "melody" {
      Non ri -- petu -- to.
      \repeat volta 2 { Ri -- petu -- to due. }
    }
  }
}

```

```

    }
  }
  >>
}

```



Le parole verranno poi espanse correttamente se le ripetizioni sono ripetute.

```

\score {
  \unfoldRepeats {
    <<
      \new Staff {
        \new Voice = "melody" {
          \relative {
            a'4 a a a
            \repeat volta 2 { b4 b b b }
          }
        }
      }
    }
  \new Lyrics {
    \lyricsto "melody" {
      Non ri -- petu -- to.
      \repeat volta 2 { Ri -- petu -- to due. }
    }
  }
  >>
}

```



Se la sezione ripetuta deve essere ripetuta di nuovo con parole diverse, è sufficiente inserire tutte le parole:

```

\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat unfold 2 { b4 b b b }
        }
      }
    }
  \new Lyrics {
    \lyricsto "melody" {
      Non ri -- petu -- to.

```

```

    Parole della prima volta.
    Parole della seconda volta.
  }
}
>>
}

```



Quando le parole che si riferiscono a una ripetizione sono diverse, le parole di ogni ripetizione devono essere inserite in contesti `Lyrics` separati, annidati correttamente in sezioni parallele:

```

\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat volta 2 { b4 b b b }
        }
      }
    }
    \new Lyrics \lyricsto "melody" {
      Non ri -- petu -- to.
    }
  <<
  { Parole della prima volta. }
  \new Lyrics {
    \set associatedVoice = "melody"
    Parole della seconda volta.
  }
  >>
}
>>
}

```



Si possono aggiungere più strofe in modo analogo:

```

\score {
  <<
    \new Staff {

```

```

\new Voice = "singleVoice" {
\relative {
  a'4 a a a
  \repeat volta 3 { b4 b b b }
  c4 c c c
}
}
}
\new Lyrics \lyricsto "singleVoice" {
  Non ri -- petu -- to.
  <<
{ Parole della prima volta. }
\new Lyrics {
  \set associatedVoice = "singleVoice"
  Parole della seconda volta.
}
\new Lyrics {
  \set associatedVoice = "singleVoice"
  Parole della terza volta.
}
}
  >>
  La sezione fi -- nale.
}
>>
}

```



Tuttavia, se questo costrutto si trova all'interno di un contesto multirigo come `ChoirStaff`, il testo della seconda e terza strofa apparirà sotto il rigo inferiore.

Per posizionarli correttamente usare `alignBelowContext`:

```

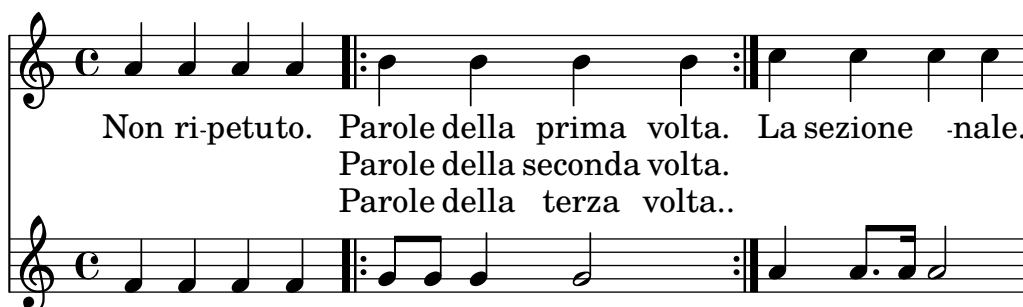
\score {
  <<
  \new Staff {
    \new Voice = "melody" {
\relative {
  a'4 a a a
  \repeat volta 3 { b4 b b b }
  c4 c c c
}
}
}
  \new Lyrics = "firstVerse" \lyricsto "melody" {
    Non ri -- petu -- to.
    <<
{ Parole della prima volta. }
\new Lyrics = "secondVerse"

```

```

\with { alignBelowContext = #"firstVerse" } {
  \set associatedVoice = "melody"
  Parole della seconda volta.
}
\new Lyrics = "thirdVerse"
\with { alignBelowContext = #"secondVerse" } {
  \set associatedVoice = "melody"
  Parole della terza volta..
}
  >>
  La sezione fi -- nale.
}
\new Voice = "harmony" {
  \relative {
f'4 f f f \repeat volta 2 { g8 g g4 g2 } a4 a8. a16 a2
  }
}
>>
}

```



## Ripetizioni con finali alternativi

Se le parole della sezione ripetuta sono le stesse, e nessuno dei finali alternativi inizia con una pausa, si può usare la stessa identica struttura sia per il testo che per la musica. Ciò comporta il vantaggio che `unfoldRepeats` espanderà correttamente sia la musica che il testo vocale.

```

\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat volta 2 { b4 b }
          \alternative { { b b } { b c } }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Non ri -- petu -- to.
        \repeat volta 2 { Ripe -- tu -- }
        \alternative { { to due. } { to due. } }
      }
    }
  }
}

```

```

    }
  >>
}

```



Ma quando la sezione ripetuta ha parole diverse o uno dei blocchi `\alternative` inizia con una pausa, non si può usare il costrutto della ripetizione per le parole e bisogna inserire manualmente i comandi `\skip` per le note delle sezioni alternative prive di un testo corrispondente.

Attenzione: non usare il trattino basso, `_`, per saltare le note, perché il trattino basso indica un melisma e fa sì che la sillaba precedente sia allineata a sinistra.

**Nota:** Il comando `\skip` deve essere seguito da un numero, ma questo numero viene ignorato se nel testo vocale la durata delle sillabe deriva dalla durata delle note in una melodia associata attraverso `\addlyrics` o `\lyricsto`. Ogni `\skip` salta una singola nota di un qualsiasi valore, senza tener conto del valore del numero che segue.

```

\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative {
          \repeat volta 2 { b'4 b }
          \alternative { { b b } { b c } }
        }
        c4 c
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Parole della prima volta.
        \repeat unfold 2 { \skip 1 }
        Termina qui.
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Pa -- role
        \repeat unfold 2 { \skip 1 }
        seconda volta.
      }
    }
  >>
}

```



Quando una nota è legata verso uno o più finali alternativi, si usa una legatura di valore per portare quella nota nel primo finale alternativo e `\repeatTie` per portarla nel secondo e nei successivi. Questa struttura comporta dei problemi di difficile allineamento quando è presente il testo; si può ottenere un risultato più accettabile aumentando la lunghezza delle sezioni alternative in modo che le note legate siano contenute interamente al loro interno.

La legatura crea un melisma nella prima alternativa, ma non nella seconda e nelle successive, dunque per allineare il testo correttamente occorre disabilitare la creazione automatica dei melismi dopo la sezione delle volte e inserire dei salti manuali.

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative {
          \set melismaBusyProperties = #'()
          \repeat volta 2 { b'4 b ~}
          \alternative { { b b } { b \repeatTie c } }
          \unset melismaBusyProperties
          c4 c
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        \repeat volta 2 { Ecco una __ }
        \alternative {
          { \skip 1 strofa }
          { \skip 1 sec }
        }
      }
      onda strofa.
    }
  >>
}
```



Se `\unfoldRepeats` precede una sezione contenente `\repeatTie`, bisogna togliere `\repeatTie` per evitare che appaiano entrambi i tipi di legatura.

Quando la sezione ripetuta ha parola diverse, non si può mettere il testo in un blocco `\repeat` e bisogna inserire manualmente i comandi `\skip`, come si è visto prima.

```
\score {
  <<
    \new Staff {
```



```

\time 2/4
\new Voice = "melody" {
\relative {
\repeat volta 2 { b'4 b ~}
\alternative { { b b } { b \repeatTie c } }
c4 c
}
}
}
\new Lyrics {
\lyricsto "melody" {
Ecco una __ strofa.
\repeat unfold 2 { \skip 1 }
}
}
\new Lyrics {
\lyricsto "melody" {
Eccone un'
\repeat unfold 2 { \skip 1 }
altra da cantare.
}
}
>>
}

```



Se si desidera mostrare gli estensori e i trattini subito prima o dopo un finale alternativo, questi vanno inseriti a mano.

```

\score {
<<
\new Staff {
\time 2/4
\new Voice = "melody" {
\relative {
\repeat volta 2 { b'4 b ~}
\alternative { { b b } { b \repeatTie c } }
c4 c
}
}
}
\new Lyrics {
\lyricsto "melody" {
Ecco una __ strofa.
\repeat unfold 2 { \skip 1 }
}
}
\new Lyrics {

```

```

        \lyricsto "melody" {
Ecco "una_"
\skip 1
"_" sec -- onda  strofa.
    }
    }
>>
}

```



Quando sia la musica che le parole differiscono, è meglio nominare i contesti della voce in cui musica e testo sono diversi e assegnare il testo a quei contesti specifici:

```
\score {
  <<
    \new Voice = "melody" {
      \relative {
        <<
          {
            \voiceOne
            e'4 e8 e
          }
          \new Voice = "splitpart" {
            \voiceTwo
            c4 c
          }
        >>
      }
      \oneVoice
      c4 c |
      c
    }
    \new Lyrics \lyricsto "melody" {
      They shall not o -- ver -- come
    }
    \new Lyrics \lyricsto "splitpart" {
      We will
    }
  >>
}
```



Nella musica corale è comune avere una voce divisa in varie misure. Il costrutto `<< {...} \\ {...} >>`, in cui due (o più) espressioni musicali sono separate dalla doppia barra inversa, potrebbe sembrare il modo giusto di impostare le voci divise. Tuttavia questo costrutto assegna **tutte** le espressioni al suo interno a **NUOVI contesti Voice**, dunque *nessun testo cantato* sarà impostato per esse perché il testo si collegherà al contesto della voce originale. Di norma, non è ciò che si desidera. Il costrutto adatto in questa situazione è il passaggio polifonico temporaneo, spiegato nella sezione *Passaggi polifonici temporanei* in [\[Single-staff polyphony\]](#), pagina [\[Single-staff polyphony\]](#).

## Polifonia con testo in comune

Quando due voci contenenti durate diverse condividono lo stesso testo cantato, allineare il testo a una delle voci può creare dei problemi nell'altra voce. Per esempio, il secondo estensore del testo nell'esempio seguente è troppo corto, perché il testo è allineato solo alla voce superiore:

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
```

```

words = \lyricmode { la __ la __ }

\new Staff <<
  \new Voice = "sopranoVoice" { \voiceOne \soprano }
  \new Voice { \voiceTwo \alto }
  \new Lyrics \lyricsto "sopranoVoice" \words
>>

```



Per ottenere il risultato desiderato, si allinea il testo a un nuovo contesto `NullVoice` contenente una giusta combinazione delle due voci. Le note del contesto `NullVoice` non appaiono nello spartito, ma servono soltanto ad allineare il testo nel modo corretto:

```

soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
aligner = \relative { b'8( c d c) b( a g a) }
words = \lyricmode { la __ la __ }

\new Staff <<
  \new Voice { \voiceOne \soprano }
  \new Voice { \voiceTwo \alto }
  \new NullVoice = "aligner" \aligner
  \new Lyrics \lyricsto "aligner" \words
>>

```



Questo metodo si può usare anche con la funzione `\partcombine`, che di per sé non accetta il testo cantato:

```

soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
aligner = \relative { b'8( c d c) b( a g a) }
words = \lyricmode { la __ la __ }

\new Staff <<
  \new Voice \partcombine \soprano \alto
  \new NullVoice = "aligner" \aligner
  \new Lyrics \lyricsto "aligner" \words
>>

```



## Problemi noti e avvertimenti

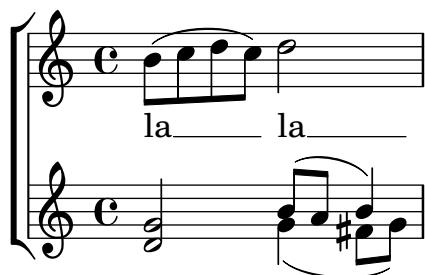
La funzione `\addLyrics` funziona solo con testi vocali collegati a contesti `Voice` e non può essere usata con `NullVoice`.

La funzione `\partcombine` è descritta in [\[Automatic part combining\]](#), pagina [\[undefined\]](#).

Infine, si può usare questo metodo anche quando le voci si trovano su righe diversi e non è limitata a due sole voci:

```
soprano = \relative { b'8( c d c) d2 }
altoOne = \relative { g'2 b8( a b4) }
altoTwo = \relative { d'2 g4( fis8 g) }
aligner = \relative { b'8( c d c) d( d d d) }
words = \lyricmode { la __ la __ }

\new ChoirStaff \with {\accepts NullVoice } <<
  \new Staff <<
    \soprano
    \new NullVoice = "aligner" \aligner
  >>
  \new Lyrics \lyricsto "aligner" \words
  \new Staff \partcombine \altoOne \altoTwo
>>
```



### 2.1.3 Strofe

#### Aggiungere i numeri di strofa

I numeri di strofa si aggiungono impostando `stanza`:

```
\new Voice \relative {
  \time 3/4 g'2 e4 a2 f4 g2.
} \addlyrics {
  \set stanza = #"1. "
  Hi, my name is Bert.
} \addlyrics {
  \set stanza = #"2. "
  Oh, ché -- ri, je t'aime
}
```



1. Hi, my name is Bert.
2. Oh, ché - ri, je t'aime

Questi numeri appaiono prima dell'inizio della prima sillaba.

## Aggiungere le dinamiche alle strofe

Le strofe che hanno un volume diverso possono essere indicate con un segno di dinamica all'inizio di ogni strofa. In LilyPond, tutto ciò che si trova di fronte a una strofa va nell'oggetto `StanzaNumber`; lo stesso vale per i segni di dinamica. Per ragioni tecniche, bisogna impostare la strofa fuori da `\lyricmode`:

```
text = {
  \set stanza = \markup { \dynamic "ff" "1. " }
  \lyricmode {
    Big bang
  }
}
```

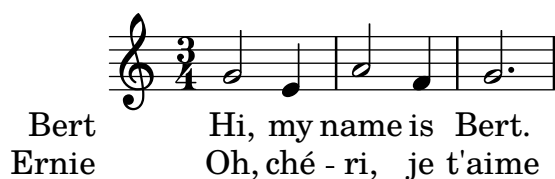
```
<<
  \new Voice = "tune" {
    \time 3/4
    g'4 c'2
  }
\new Lyrics \lyricsto "tune" \text
>>
```



## Aggiungere i nomi dei cantanti alle strofe

Si possono aggiungere anche i nomi dei cantanti. Appariranno all'inizio del rigo, proprio come per i nomi degli strumenti. Si creano impostando `vocalName`. Una versione abbreviata si inserisce con `shortVocalName`.

```
\new Voice \relative {
  \time 3/4 g'2 e4 a2 f4 g2.
} \addlyrics {
  \set vocalName = #"Bert "
  Hi, my name is Bert.
} \addlyrics {
  \set vocalName = #"Ernie "
  Oh, ché -- ri, je t'aime
}
```



## Strofe con durate diverse

Spesso, strofe diverse di una canzone sono collegate a una melodia in modi leggermente diversi. Tali variazioni possono essere colte con `\lyricsto`.

## Ignorare i melismi

Può capitare ad esempio che il testo abbia un melisma in una strofa, ma varie sillabe in un'altra. Una possibile soluzione consiste nel far sì che la voce più veloce ignori il melisma, impostando `ignoreMelismata` nel contesto `Lyrics`.

```
<<
\relative \new Voice = "lahlah" {
  \set Staff.autoBeaming = ##f
  c'4
  \slurDotted
  f8.[( g16)]
  a4
}
\new Lyrics \lyricsto "lahlah" {
  più len -- ta
}
\new Lyrics \lyricsto "lahlah" {
  più
  \set ignoreMelismata = ##t
  velo -- ce
  \unset ignoreMelismata
  ancora
}
>>
```



## Problemi noti e avvertimenti

Diversamente dalla maggior parte dei comandi `\set`, `\set ignoreMelismata` non funziona se preceduto da `\once`. Bisogna usare `\set` e `\unset` per contrassegnare il testo in cui il melisma deve essere ignorato.

## Aggiungere le sillabe agli abbellimenti

Per impostazione predefinita, gli abbellimenti (ovvero le note inserite con `\grace`) non sono assegnati alle sillabe quando si usa `\lyricsto`, ma tale comportamento può essere modificato:

```
<<
\new Voice = melody \relative {
  f'4 \appoggiatura a32 b4
  \grace { f16 a16 } b2
  \afterGrace b2 { f16[ a16] }
  \appoggiatura a32 b4
  \acciaccatura a8 b4
}
\new Lyrics
\lyricsto melody {
  normal
  \set includeGraceNotes = ##t
}
```

```

    case,
    gra -- ce case,
    after -- grace case,
    \set ignoreMelismata = ##t
    app. case,
    acc. case.
  }
>>

```



## Problemi noti e avvertimenti

Come per `associatedVoice`, `includeGraceNotes` deve essere impostato al più tardi una sillaba prima di quella da mettere sotto un abbellimento. Per il caso di un abbellimento proprio all'inizio di un brano, meglio usare un blocco `\with` o `\context`:

```

<<
  \new Voice = melody \relative c' {
    \grace { c16( d e f }
    g1) f
  }
  \new Lyrics \with { includeGraceNotes = ##t }
  \lyricsto melody {
    Ah __ fa
  }
>>

```



## Passare a una melodia alternativa

Sono possibili variazioni più complesse nell'impostare testo e musica. La melodia su cui è impostato il testo può essere modificata all'interno del contesto del testo impostando la proprietà `associatedVoice`:

```

<<
  \relative \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
    c'4
    <<
      \new Voice = "alternative" {
        \voiceOne
        \tuplet 3/2 {
          % mostra chiaramente le associazioni.
          \override NoteColumn.force-hshift = #-3
          f8 f g
        }
      }
    }
  }
>>

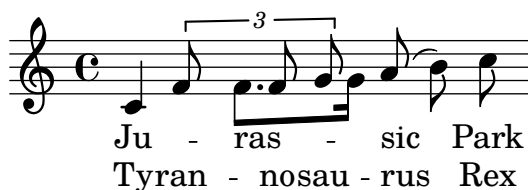
```



```

    }
    {
\voiceTwo
f8.[ g16]
\oneVoice
    } >>
    a8( b) c
}
\new Lyrics \lyricsto "lahlah" {
  Ju -- ras -- sic Park
}
\new Lyrics \lyricsto "lahlah" {
  % Complicato: bisogna impostare associatedVoice
  % una sillaba prima di quella cui si applica!
  \set associatedVoice = "alternative" % si applica a "ran"
  Ty --
  ran --
  no --
  \set associatedVoice = "lahlah" % si applica a "rus"
  sau -- rus Rex
} >>

```



Il testo per la prima strofa viene impostato sulla melodia 'lahlah' nel solito modo, ma la seconda strofa è impostata inizialmente sul contesto `lahlah` e passa poi alla melodia `alternative` per le sillabe da 'ran' a 'sau':

```

\set associatedVoice = "alternative" % si applica a "ran"
Ty --
ran --
no --
\set associatedVoice = "lahlah" % si applica a "rus"
sau -- rus Rex

```

In questo esempio `alternative` è il nome del contesto `Voice` contenente la terzina.

Attenzione al posizionamento del comando `\set associatedVoice`: appare una sillaba troppo presto, ma ciò è corretto.

**Nota:** Il comando `\set associatedVoice` deve essere inserito una sillaba *prima* di quella in cui deve verificarsi il passaggio alla nuova voce. In altre parole, il passaggio alla voce associata accade una sillaba dopo quella che ci si aspetterebbe. Ciò è dovuto a ragioni tecniche e non è un difetto di LilyPond.

## Stampare le strofe alla fine

Talvolta si allinea una sola strofa alla musica e le strofe rimanenti appaiono in forma di versi alla fine del brano. Per ottenere ciò si aggiungono le strofe ulteriori in un blocco `\markup` esterno al

blocco della partitura. Nell'esempio seguente si possono notare due modi di forzare le interruzioni di linea in un blocco `\markup`.

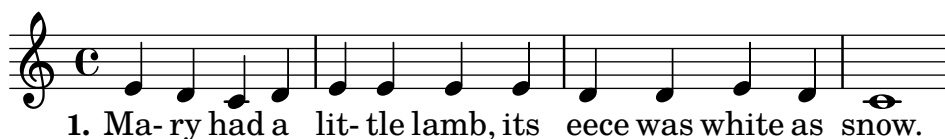
```
melody = \relative {
e' d c d | e e e e |
d d e d | c1 |
}

text = \lyricmode {
\set stanza = #"1." Ma- ry had a lit- tle lamb,
its fleece was white as snow.
}

\score{ <<
  \new Voice = "one" { \melody }
  \new Lyrics \lyricsto "one" \text
>>
  \layout { }
}
\markup { \column{
  \line{ Verse 2. }
  \line{ All the children laughed and played }
  \line{ To see a lamb at school. }
}
}
\markup{
  \wordwrap-string #"
  Verse 3.

  Mary took it home again,

  It was against the rule."
}
```



Verse 2.  
All the children laughed and played  
To see a lamb at school.

Verse 3.  
Mary took it home again,  
It was against the rule.

## Stampare le strofe alla fine in molteplici colonne

Quando un brano ha molte strofe, queste sono spesso stampate in molteplici colonne lungo la pagina. Un numero di strofa rientrato spesso introduce ciascuna strofa. L'esempio seguente mostra come riprodurre questo output in LilyPond.

```
melody = \relative {
```

```

    c'4 c c c | d d d d
}

text = \lyricmode {
  \set stanza = #"1."
  This is verse one.
  It has two lines.
}

\score {
  <<
    \new Voice = "one" { \melody }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
}

\markup {
  \fill-line {
    \hspace #0.1 % sposta la colonna in avanti rispetto al margine sinistro;
    % può essere tolto se lo spazio nella pagina è stretto
    \column {
      \line { \bold "2."
    \column {
      "This is verse two."
      "It has two lines."
    }
  }
  \combine \null \vspace #0.1 % aggiunge spazio verticale tra le strofe
  \line { \bold "3."
\column {
  "This is verse three."
  "It has two lines."
}
}
}
  \hspace #0.1 % aggiunge spazio orizzontale tra le colonne;
  \column {
    \line { \bold "4."
  \column {
    "This is verse four."
    "It has two lines."
  }
}
  \combine \null \vspace #0.1 % aggiunge spazio verticale tra le strofe
  \line { \bold "5."
\column {
  "This is verse five."
  "It has two lines."
}
}
}
}

```

```
\hspace #0.1 % dà ulteriore spazio sul margine destro;
% può essere tolto se lo spazio nella pagina è stretto
}
```



1. This is verse one. It has two lines.

2. This is verse two.

It has two lines.

3. This is verse three.

It has two lines.

4. This is verse four.

It has two lines.

5. This is verse ve.

It has two lines.

## Vedi anche

Guida al funzionamento interno: Sezione “LyricText” in *Guida al Funzionamento Interno*, Sezione “StanzaNumber” in *Guida al Funzionamento Interno*.

### 2.1.4 Canzoni

#### Riferimenti per canzoni

Le canzoni si scrivono solitamente su tre righe: la melodia per il cantante nel rigo superiore e due righe per l’accompagnamento di pianoforte in basso. Il testo della prima strofa appare immediatamente sotto il rigo superiore. Se ci sono solo poche altre strofe, queste possono essere poste subito sotto la prima; ma se ci sono più strofe di quante ne possano essere contenute lì la seconda strofa e le successive vengono stampate dopo la musica come testo separato.

Tutti gli elementi della notazione necessari per scrivere canzoni sono descritti dettagliatamente in altre parti della documentazione:

- Per costruire la struttura del rigo: `\displaying staves`, pagina `\displaying staves`.
- Per scrivere la musica per pianoforte: Sezione 2.2 [Keyboard and other multi-staff instruments], pagina 324.
- Per scrivere il testo da associare a una linea melodica: `\common notation for vocal music`, pagina `\common notation for vocal music`.
- Per posizionare il testo: `\placing lyrics vertically`, pagina `\placing lyrics vertically`.
- Per inserire le strofe: `\stanzas`, pagina `\stanzas`.
- Le canzoni hanno spesso i nomi degli accordi che appaiono sopra i righe. Questo argomento è trattato in Sezione 2.7.2 [Displaying chords], pagina 416.
- Per stampare i diagrammi degli accordi per l’accompagnamento per chitarra o altri strumenti a tasti: “Fret diagram markups” in Sezione 2.4.1 [Common notation for fretted strings], pagina 339.

## Vedi anche

Manuale di apprendimento: Sezione “Canzoni” in *Manuale di Apprendimento*.

Guida alla notazione: `\common notation for vocal music`, pagina `\common notation for vocal music`, Sezione 2.7.2 [Displaying chords], pagina 416, `\displaying staves`, pagina `\displaying staves`, Sezione 2.2 [Keyboard and other multi-staff instruments], pagina 324, `\placing lyrics vertically`, pagina `\placing lyrics vertically`, `\stanzas`, pagina `\stanzas`.

Frammenti: Sezione “Vocal music” in *Frammenti di codice*.

## Canzonieri

I canzonieri (in inglese *lead sheet*) si ottengono combinando le parti vocali con la ‘modalità per accordi’; la sintassi è spiegata in Sezione 2.7 [Chord notation], pagina 411.

### Frammenti di codice selezionati

#### *Canzoniere semplice*

Mettendo insieme nomi degli accordi, melodia e testo si ottiene un canzoniere (in inglese “lead sheet”):

```
<<
\chords { c2 g:sus4 f e }
\relative c'' {
  a4 e c8 e r4
  b2 c4( d)
}
\addlyrics { One day this shall be free __ }
>>
```



## Vedi anche

Guida alla notazione: Sezione 2.7 [Chord notation], pagina 411.

### 2.1.5 Musica corale

Questa sezione tratta le questioni che hanno a che fare più direttamente con la musica corale, compresi inni, cori polifonici, oratori, etc.

### Riferimenti per musica corale

La musica corale viene solitamente rappresentata con due, tre o quattro righe in un gruppo **ChoirStaff**. L’accompagnamento, se richiesto, è posto sotto in un gruppo **PianoStaff**, che viene solitamente rimpicciolito per le prove di opere corali *a cappella*. Le note di ogni parte vocale sono inserite in un contesto **Voice** e a ogni rigo viene assegnato una sola parte vocale (ovvero un contesto **Voice**) o una coppia di parti vocali (due contesti **Voice**).

Le parole vengono poste nei contesti **Lyrics**, o sotto ciascun rigo musicale corrispondente oppure una strofa sopra e una sotto il rigo musicale se questo contiene musica per due parti.

Vari argomenti comuni nella musica corale sono trattati dettagliatamente in altre sezioni della documentazione:

- Un’introduzione alla creazione di una partitura vocale SATB si trova nel manuale di apprendimento: Sezione “Partitura vocale a quattro parti SATB” in *Manuale di Apprendimento*. È disponibile anche un modello integrato che semplifica la scrittura di musica vocale SATB: Sezione “Modelli integrati” in *Manuale di Apprendimento*.
- Vari modelli adatti per vari stili di musica corale si trovano anche nel manuale di apprendimento: Sezione “Modelli per gruppi vocali” in *Manuale di Apprendimento*.
- Le informazioni relative a **ChoirStaff** e **PianoStaff** si trovano in [\[Grouping staves\]](#), pagina [\[undefined\]](#).

- Le teste di nota a forma variabile, come quelle usate nello stile Sacred Harp e notazione simile, sono descritte in `<undefined>` [Shape note heads], pagina `<undefined>`.
- Quando due parti vocali si trovano su uno stesso rigo, i gambi, le legature, etc. della parte più alta sono dirette in su e quelle della parte più bassa sono rivolte in giù. Per farlo si usa `\voiceOne` e `\voiceTwo`, come è spiegato in `<undefined>` [Single-staff polyphony], pagina `<undefined>`.
- Quando una parte vocale si divide temporaneamente, si devono usare i *passaggi polifonici temporanei* (vedi `<undefined>` [Single-staff polyphony], pagina `<undefined>`).

## Comandi predefiniti

`\oneVoice`, `\voiceOne`, `\voiceTwo`.

## Vedi anche

Manuale di apprendimento: Sezione “Partitura vocale a quattro parti SATB” in *Manuale di Apprendimento*, Sezione “Modelli per gruppi vocali” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.1.7 [Context layout order], pagina 596, `<undefined>` [Grouping staves], pagina `<undefined>`, `<undefined>` [Shape note heads], pagina `<undefined>`, `<undefined>` [Single-staff polyphony], pagina `<undefined>`.

Frammenti: Sezione “Vocal music” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “ChoirStaff” in *Guida al Funzionamento Interno*, Sezione “Lyrics” in *Guida al Funzionamento Interno*, Sezione “PianoStaff” in *Guida al Funzionamento Interno*.

## Struttura di una partitura corale

La musica corale contenente quattro righe, con o senza accompagnamento del pianoforte, viene solitamente disposta in due sistemi per pagina. A seconda della dimensione della pagina, ottenere ciò può richiedere modifiche a varie impostazioni predefinite. Occorre considerare le seguenti impostazioni:

- La dimensione globale del rigo può essere modificata per cambiare la dimensione degli elementi della partitura. Vedi `<undefined>` [Setting the staff size], pagina `<undefined>`.
- Le distanze tra i sistemi, i righe e il testo cantato possono essere tutte regolate in modo indipendente. Vedi `<undefined>` [Vertical spacing], pagina `<undefined>`.
- Le dimensioni delle variabili di disposizione verticale possono essere mostrate per facilitare la regolazione della spaziatura verticale. Questa e altre possibilità per far entrare la musica in meno pagine sono descritte in `<undefined>` [Fitting music onto fewer pages], pagina `<undefined>`.
- Se il numero dei sistemi per pagina cambia da uno a due, è uso comune indicare ciò con un separatore di sistema tra i due sistemi. Vedi `<undefined>` [Separating systems], pagina `<undefined>`.
- Per maggiori informazioni sulle proprietà di formattazione della pagina, leggere `<undefined>` [Page layout], pagina `<undefined>`.

Le indicazioni dinamiche sono poste, per impostazione predefinita, sotto il rigo, ma nella musica corale sono poste solitamente sopra il rigo per evitare il testo cantato. Il comando predefinito `\dynamicUp` permette di ottenere ciò per le indicazioni dinamiche in un singolo contesto **Voice**. Se ci sono molti contesti **Voice**, questo comando dovrebbe essere posto in ciascuno di essi. Ma esiste la possibilità di usare una sola volta la sua forma estesa, che sposta tutte le indicazioni dinamiche dell'intera partitura sopra i loro rispettivi righe, come mostrato in questo esempio:

```
\score {
```

```

\new ChoirStaff <<
  \new Staff {
    \new Voice {
\relative { g'4\ff g g g }
    }
  }
  \new Staff {
    \new Voice {
\relative { d'4 d d\p d }
    }
  }
>>
\layout {
  \context {
    \Score
    \override DynamicText.direction = #UP
    \override DynamicLineSpanner.direction = #UP
  }
}

```



## Comandi predefiniti

\dynamicUp, \dynamicDown, \dynamicNeutral.

## Vedi anche

Guida alla notazione: [\[Changing spacing\]](#), pagina [\[Displaying spacing\]](#), pagina [\[Fitting music onto fewer pages\]](#), pagina [\[Page layout\]](#), pagina [\[Score layout\]](#), pagina [\[Separating systems\]](#), pagina [\[Setting the staff size\]](#), pagina [\[Breaks\]](#), pagina [\[Vertical spacing\]](#), pagina [\[Vertical spacing\]](#).

Guida al funzionamento interno: Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*, Sezione “StaffGrouper” in *Guida al Funzionamento Interno*.

## Voci divise

*Usare arpeggioBracket per rendere i divisi più visibili*

Si può usare `arpeggioBracket` per indicare la divisione delle voci quando non ci sono gambi che forniscano questa informazione. Questo caso è frequente nella musica corale.

```
\include "english.ly"
```

```

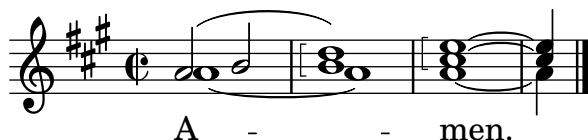
\score {
  \relative c'' {

```

```

\key a \major
\time 2/2
<<
  \new Voice = "upper"
  <<
    { \voiceOne \arpeggioBracket
      a2( b2
      <b d>1\arpeggio)
      <cs e>\arpeggio ~
      <cs e>4
    }
    \addlyrics { \lyricmode { A -- men. } }
  >>
  \new Voice = "lower"
  { \voiceTwo
    a1 ~
    a
    a ~
    a4 \bar "|."
  }
>>
}
\layout { ragged-right = ##t }
}

```



## Vedi anche

Guida alla notazione: [\[Expressive marks as lines\]](#), pagina [\[undefined\]](#).

### 2.1.6 Opera e musical

La musica, il testo cantato e i dialoghi delle opere e dei musical sono di solito impostati in una o più delle seguenti forme:

- Una *partitura del direttore* contenente l'intera partitura orchestrale e le parti vocali, insieme a un libretto con le battute d'entrata se ci sono dei passaggi parlati.
- Le *parti orchestrali* contenenti la musica degli strumenti individuali dell'orchestra o della band.
- Una *partitura vocale* contenente tutte le parti vocali con accompagnamento del pianoforte. L'accompagnamento di solito è una riduzione orchestrale e in questo caso il nome dello strumento orchestrale originale viene spesso indicato. Le partiture vocali talvolta hanno anche le didascalie e il libretto con le battute d'entrata.
- Un *libro vocale* contenente soltanto le parti vocali (nessun accompagnamento), talvolta insieme al libretto.
- Un *libretto* contenente i passaggi estesi dei dialoghi parlati solitamente presenti nei musical, insieme alle parole delle parti cantate. Le didascalie sono solitamente incluse. È possibile usare LilyPond per creare i libretti, ma dato che non contengono musica è preferibile usare altri metodi.



Le sezioni della documentazione di LilyPond che trattano gli argomenti necessari per creare partiture negli stili più comuni nell'opera e nei musical sono indicati nei riferimenti seguenti. A questi seguono sezioni che trattano le tecniche specifiche che sono peculiari per le partiture di opera e musical.

## Riferimenti per opera e musical

- La partitura di un direttore d'orchestra contiene molti righi e testi vocali raggruppati. I modi per raggruppare i righi sono mostrati in [\[Grouping staves\]](#), pagina [\[Grouping staves\]](#). Per annidare gruppi di righi leggere [\[Nested staff groups\]](#), pagina [\[Nested staff groups\]](#).
- La stampa di righi vuoti nelle partiture musicali e vocali dei direttori è solitamente soppressa. Per creare tale partitura leggere [\[Hiding staves\]](#), pagina [\[Hiding staves\]](#).
- La scrittura di parti orchestrali è trattata in [\[Writing parts\]](#), pagina [\[Writing parts\]](#). Altre sezioni del capitolo Notazione specialistica potrebbero essere rilevanti, a seconda dell'orchestrazione usata. Molti strumenti sono strumenti traspositori, vedi [\[Instrument transpositions\]](#), pagina [\[Instrument transpositions\]](#).
- Se il numero di sistemi per pagina cambia da pagina a pagina, è di uso comune separare i sistemi con un segno separatore di sistemi. Vedi [\[Separating systems\]](#), pagina [\[Separating systems\]](#).
- Per maggiori informazioni sulle proprietà di formattazione della pagina leggere [\[Page layout\]](#), pagina [\[Page layout\]](#).
- Si possono inserire suggerimenti di dialogo, didascalie e note a piè di pagina, vedi [\[Creating footnotes\]](#), pagina [\[Creating footnotes\]](#), e [\[Text\]](#), pagina [\[Text\]](#). Didascalie estese possono essere aggiunte anche con una sezione di markup indipendenti tra i due blocchi `\score`, vedi [\[Separate text\]](#), pagina [\[Separate text\]](#).

## Vedi anche

Glossario musicale: Sezione “Partitura senza righi vuoti” in *Glossario Musicale*, Sezione “Rigo temporaneo” in *Glossario Musicale*, Sezione “Strumento traspositore” in *Glossario Musicale*.

Guida alla notazione: [\[Creating footnotes\]](#), pagina [\[Creating footnotes\]](#), [\[Grouping staves\]](#), pagina [\[Grouping staves\]](#), [\[Hiding staves\]](#), pagina [\[Hiding staves\]](#), [\[Instrument transpositions\]](#), pagina [\[Instrument transpositions\]](#), [\[Nested staff groups\]](#), pagina [\[Nested staff groups\]](#), [\[Page layout\]](#), pagina [\[Page layout\]](#), [\[Separating systems\]](#), pagina [\[Separating systems\]](#), [\[Transpose\]](#), pagina [\[Transpose\]](#), [\[Writing parts\]](#), pagina [\[Writing parts\]](#), [\[Writing text\]](#), pagina [\[Writing text\]](#).

Frammenti: Sezione “Vocal music” in *Frammenti di codice*.

## Nomi dei personaggi

I nomi dei personaggi sono solitamente mostrati a sinistra del rigo quando il rigo è dedicato a quel personaggio soltanto:

```
\score {
  <<
    \new Staff {
      \set Staff.vocalName = \markup \smallCaps Kaspar
      \set Staff.shortVocalName = \markup \smallCaps Kas.
      \relative {
        \clef "G_8"
        c'4 c c c
        \break
        c4 c c c
      }
    }
}
```

```

    }
  }
  \new Staff {
    \set Staff.vocalName = \markup \smallCaps Melchior
    \set Staff.shortVocalName = \markup \smallCaps Mel
    \clef "bass"
    \relative {
a4 a a a
a4 a a a
    }
  }
}
>>
}

```



Quando due o più personaggi condividono lo stesso rigo, il nome del personaggio è solitamente collocato sopra il rigo all'inizio di ogni sezione appartenente a quel personaggio. È possibile fare ciò con i markup. Spesso si usa un tipo di carattere preciso a questo scopo.

```

\relative c' {
  \clef "G_8"
  c4~\markup \fontsize #1 \smallCaps Kaspar
  c c c
  \clef "bass"
  a4~\markup \fontsize #1 \smallCaps Melchior
  a a a
  \clef "G_8"
  c4~\markup \fontsize #1 \smallCaps Kaspar
  c c c
}

```



Altrimenti, se ci sono molti cambi di personaggi, è più semplice impostare una variabile per salvare le definizioni di ogni personaggio, in modo che il cambio di personaggio possa essere indicato in modo facile e conciso.

```

kaspar = {

```

```

\clef "G_8"
\set Staff.shortVocalName = "Kas."
\set Staff.midiInstrument = "voice oohs"
<>^\markup \smallCaps "Kaspar"
}

melchior = {
  \clef "bass"
  \set Staff.shortVocalName = "Mel."
  \set Staff.midiInstrument = "choir aahs"
  <>^\markup \smallCaps "Melchior"
}

\relative c' {
  \kaspar
  c4 c c c
  \melchior
  a4 a a a
  \kaspar
  c4 c c c
}

```



## Vedi anche

Manuale di apprendimento: Sezione “Organizzare i brani con le variabili” in *Manuale di Apprendimento*.

Guida alla notazione: [\[Text\]](#), pagina [\[Text markup commands\]](#), pagina [\[Text markup commands\]](#).

## Suggerimenti musicali

I suggerimenti musicali possono essere inseriti nelle partiture vocali, nei libri vocali e nelle parti orchestrali per indicare quale musica in un'altra parte precede immediatamente un'entrata. I suggerimenti sono spesso inseriti anche nella riduzione per pianoforte nelle partiture vocali per indicare cosa sta suonando ogni strumento dell'orchestra. Ciò aiuta il direttore quando non è disponibile la partitura completa.

Il meccanismo di base per inserire i suggerimenti è spiegato dettagliatamente in [\[Quoting other voices\]](#), pagina [\[Quoting other voices\]](#), e [\[Formatting cue notes\]](#), pagina [\[Formatting cue notes\]](#). Ma quando si devono inserire molti suggerimenti, per esempio per aiutare il direttore in una partitura vocale, il nome dello strumento deve essere posizionato attentamente proprio prima dell'inizio delle citazioni in corpo più piccolo (“cue notes”). L'esempio seguente mostra come si fa.

```

flute = \relative {
  s4 s4 e' ' g
}
\addQuote "flute" { \flute }

pianoRH = \relative {

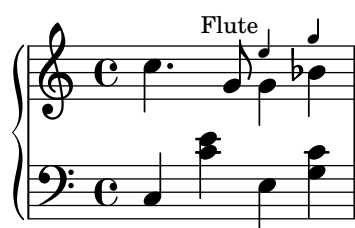
```

```

c''4. g8
% posiziona il nome dello strumento citato proprio prima delle citazioni in corpo piccolo
% e sopra il rigo
<>^\markup { \right-align { \tiny "Flute" } }
\cueDuring "flute" #UP { g4 bes4 }
}
pianoLH = \relative { c4 <c' e> e, <g c> }

\score {
  \new PianoStaff <<
    \new Staff {
      \pianoRH
    }
    \new Staff {
      \clef "bass"
      \pianoLH
    }
  >>
}

```



Se viene citato uno strumento traspositore, la parte strumentale deve specificare la sua armatura di chiave in modo che la conversione delle sue notine sia fatta automaticamente. Il prossimo esempio mostra questa trasposizione per un clarinetto in Si bemolle. Le note in questo esempio si trovano in basso nel rigo, quindi viene specificato DOWN in `\cueDuring` (in modo che i gambi vadano giù) e il nome dello strumento è posizionato sotto il rigo.

```

clarinet = \relative c' {
  \transposition bes
  fis4 d d c
}
\addQuote "clarinet" { \clarinet }

pianoRH = \relative c'' {
  \transposition c'
  % posiziona il nome dello strumento citato sotto il rigo
  <>_\markup { \right-align { \tiny "Clar." } }
  \cueDuring "clarinet" #DOWN { c4. g8 }
  g4 bes4
}
pianoLH = \relative { c4 <c' e> e, <g c> }

\score {
  <<
    \new PianoStaff <<
      \new Staff {

```

```

\new Voice {
  \pianoRH
}
}
\new Staff {
\clef "bass"
\pianoLH
}
>>
>>
}

```



Da questi due esempi è evidente che inserire molte citazioni in corpo piccolo in una partitura vocale sarebbe noioso, e le note della parte per pianoforte sarebbero confuse. Tuttavia, come mostra il frammento seguente, è possibile definire una funzione musicale per ridurre l'input e rendere più chiare le note per pianoforte.

## Frammenti di codice selezionati

### *Aggiungere citazioni orchestrali a una partitura vocale*

L'esempio seguente mostra un approccio per semplificare l'aggiunta di citazioni orchestrali a una riduzione per pianoforte di una partitura vocale. La funzione musicale `\cueWhile` prende quattro argomenti: la musica da cui prendere la citazione, come è definita da `\addQuote`, il nome da inserire prima delle notine, poi o `#UP` o `#DOWN` per specificare o `\voiceOne` col nome sopra il rigo o `\voiceTwo` col nome sotto il rigo, e infine la musica per pianoforte che deve apparire in parallelo alle notine. Il nome dello strumento citato è posto a sinistra delle notine. Molti passaggi possono essere citati, ma non possono sovrapporsi l'un l'altro nel tempo.

```

cueWhile =
#(define-music-function
  (instrument name dir music)
  (string? string? ly:dir? ly:music?)
  #{
    \cueDuring $instrument #dir {
      \once \override TextScript.self-alignment-X = #RIGHT
      \once \override TextScript.direction = $dir
      <->\markup { \tiny #name }
      $music
    }
  })

flute = \relative c'' {
  \transposition c'
  s4 s4 e g
}
\addQuote "flute" { \flute }

```

```

clarinet = \relative c' {
  \transposition bes
  fis4 d d c
}
\addQuote "clarinet" { \clarinet }

singer = \relative c'' { c4. g8 g4 bes4 }
words = \lyricmode { here's the lyr -- ics }

pianoRH = \relative c'' {
  \transposition c'
  \cueWhile "clarinet" "Clar." #DOWN { c4. g8 }
  \cueWhile "flute" "Flute" #UP { g4 bes4 }
}
pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  <<
    \new Staff {
      \new Voice = "singer" {
        \singer
      }
    }
    \new Lyrics {
      \lyricsto "singer"
      \words
    }
    \new PianoStaff <<
      \new Staff {
        \new Voice {
          \pianoRH
        }
      }
      \new Staff {
        \clef "bass"
        \pianoLH
      }
    >>
  >>
}

```



## Vedi anche

Glossario musicale: Sezione “Notine o Citazioni in corpo più piccolo” in *Glossario Musicale*.

Guida alla notazione: Sezione 5.5.1 [Aligning objects], pagina 627, Sezione 5.4.2 [Direction and placement], pagina 611, [\[Formatting cue notes\]](#), pagina [\[Quoting other voices\]](#), pagina [\[Using music functions\]](#), pagina [\[Using music functions\]](#).

Frammenti: Sezione “Vocal music” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “CueVoice” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

`\cueDuring` inserisce automaticamente un contesto `CueVoice` e tutte le notine sono poste in quel contesto. Ciò significa che non è possibile avere due sequenze sovrapposte di notine con questa tecnica. Si possono inserire sequenze sovrapposte solo dichiarando esplicitamente contesti `CueVoice` distinti e usando `\quoteDuring` per estrarre e inserire le notine.

## Musica parlata

Effetti come il ‘parlato’ o ‘Sprechgesang’ chiedono all’esecutore di parlare senza intonare note ma andando comunque a ritmo; tali effetti si indicano con teste di nota barrate, come è illustrato in [\[Special note heads\]](#), pagina [\[Special note heads\]](#).

## Dialogo sopra la musica

Il dialogo parallelo alla musica appare solitamente sopra i righi in corsivo, con l’inizio di ogni frase collegato a un momento musicale ben preciso.

In caso di brevi intromissioni può bastare un semplice `\markup`.

```
\relative {
  a'4^\markup { \smallCaps { Alex - } \italic { He's gone } } a a a
  a4 a a^\markup { \smallCaps { Bethan - } \italic { Where? } } a
  a4 a a a
}
```



In caso di frasi più lunghe può essere necessario espandere la musica per poter far entrare le parole. Non c’è modo di fare ciò del tutto automaticamente in LilyPond e sarà necessario anche qualche intervento manuale per formattare la pagina.

In caso di frasi lunghe o di passaggi con molti dialoghi serrati, l’uso di un contesto `Lyrics` darà risultati migliori. Il contesto `Lyrics` non deve essere associato a una voce musicale; occorre invece assegnare a ogni parte del dialogo una durata esplicita. Se c’è un vuoto nel dialogo, la parola

finale deve essere separata dal resto e la durata divisa tra le due così che la musica sottostante abbia spazio sufficiente.

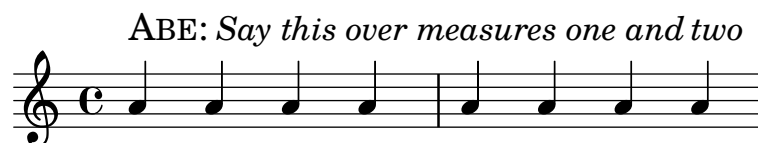
Se il dialogo si estende per più di un rigo sarà necessario inserire manualmente dei `\break` e aggiustare il posizionamento del dialogo per evitare di entrare nel margine destro. La parola finale dell'ultima misura di un rigo deve anche essere separata, come mostrato prima.

Ecco un esempio che illustra come fare.

```
music = \relative {
  \repeat unfold 3 { a'4 a a a }
}

dialogue = \lyricmode {
  \markup {
    \fontsize #1 \upright \smallCaps Abe:
    "Say this over measures one and"
  }4*7
  "two"4 |
  \break
  "and this over measure"4*3
  "three"4 |
}

\score {
  <<
    \new Lyrics \with {
      \override LyricText.font-shape = #'italic
      \override LyricText.self-alignment-X = #LEFT
    }
    { \dialogue }
    \new Staff {
      \new Voice { \music }
    }
  >>
}
```



## Vedi anche

Guida alla notazione: [Manual syllable durations](#), pagina [Manual syllable durations](#), [Manual syllable durations](#) [Text], pagina [Manual syllable durations](#).

Internal Reference: Sezione “LyricText” in *Guida al Funzionamento Interno*.



### 2.1.7 Canti salmi e inni

La musica e le parole per canti, salmi e inni di solito segue un formato ben definito in ciascuna chiesa. Sebbene i formati possano differire da chiesa a chiesa, i problemi tipografici che si possono incontrare sono generalmente simili e sono trattati in questa sezione.

#### Riferimenti per canti e salmi

La composizione tipografica dei canti gregoriani in vari stili di notazione antica è descritta in Sezione 2.9 [Ancient notation], pagina 431.

#### Vedi anche

Guida alla notazione: Sezione 2.9 [Ancient notation], pagina 431.

Frammenti: Sezione “Vocal music” in *Frammenti di codice*.

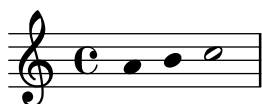
#### Impostare un canto

Le impostazioni per il canto moderno usano la notazione moderna insieme ad alcuni elementi della notazione antica. Alcuni di questi elementi e metodi sono mostrati in questa sezione.

I canti usano spesso note di un quarto senza gambi per indicare l’altezza, mentre le durate vengono dal ritmo parlato delle parole.

```
stemOff = { \hide Staff.Stem }
```

```
\relative c' {
  \stemOff
  a'4 b c2 |
}
```



Nei canti le stanghette sono spesso omesse oppure si usano delle stanghette più brevi o punteggiate per indicare le pause nella musica. Per omettere tutte le stanghette da tutti i righi si disattiva l’incisore delle stanghette:

```
\score {
  \new StaffGroup <<
    \new Staff {
      \relative {
        a'4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
    \new Staff {
      \relative {
        a'4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
  >>
}
```

```

\layout {
  \context {
    \Staff
    \remove "Bar_engraver"
  }
}

```



Le stanghette possono anche essere tolte solo in certi righi:

```

\score {
  \new ChoirStaff <<
    \new Staff
    \with { \remove "Bar_engraver" } {
      \relative {
        a'4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
    \new Staff {
      \relative {
        a'4 b c2 |
        a4 b c2 |
        a4 b c2 |
      }
    }
  >>
}

```



Per togliere le stanghette da una sezione musicale soltanto basta trattarla come una cadenza. Se la sezione è lunga potrebbe essere necessario inserire delle stanghette fittizie con `\bar ""` per mostrare dove si deve interrompere la linea.

```

\relative a' {
  a4 b c2 |
  \cadenzaOn
  a4 b c2
  a4 b c2
}

```

```

\bar ""
a4 b c2
a4 b c2
\cadenza0ff
a4 b c2 |
a4 b c2 |
}

```



Nei canti le pause si indicano con stanghette modificate.

```
\relative a' {
  a4
  \cadenza0n
  b c2
  a4 b c2
  \bar " "
  a4 b c2
  a4 b c2
  \bar " ; "
  a4 b c2
  \bar " ! "
  a4 b c2
  \bar " | | "
}
```



Altrimenti, talvolta si usa la notazione usata nel canto gregoriano per le pause anche se il resto della notazione è moderna. Nell'esempio seguente si usa un segno `\breathe` modificato:

```

divisioMinima = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::divisio-minima
  \once \override BreathingSign.Y-offset = #0
  \breathe
}

divisioMaior = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::divisio-maior
  \once \override BreathingSign.Y-offset = #0
  \breathe
}

divisioMaxima = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::divisio-maxima
  \once \override BreathingSign.Y-offset = #0
  \breathe
}

finalis = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::finalis
  \once \override BreathingSign.Y-offset = #0
}

```

```

\breathe
}

\score {
  \relative {
    g'2 a4 g
    \divisioMinima
    g2 a4 g
    \divisioMaior
    g2 a4 g
    \divisioMaxima
    g2 a4 g
    \finalis
  }
  \layout {
    \context {
      \Staff
      \remove "Bar_engraver"
    }
  }
}

```



Nei canti viene solitamente omessa l'indicazione di tempo e spesso anche la chiave.

```

\score {
  \new Staff {
    \relative {
      a'4 b c2 |
      a4 b c2 |
      a4 b c2 |
    }
  }
  \layout {
    \context {
      \Staff
      \remove "Bar_engraver"
      \remove "Time_signature_engraver"
      \remove "Clef_engraver"
    }
  }
}

```



I canti per salmi della tradizione anglicana sono solitamente o *singoli*, con 7 battute musicali, oppure *doppi*, con due gruppi di 7 battute. Ogni gruppo di 7 battute è diviso a metà, che corrispondono alle metà di ciascun verso, di solito separato da una doppia stanghetta. Si usano

solo semibrevi e minime. La prima battuta di ogni metà contiene sempre un solo accordo di semibrevi, che viene chiamato la “nota recitativa”. I canti sono centrati sulla pagina.

```
SopranoMusic = \relative {
  g'1 | c2 b | a1 | \bar "||"
  a1 | d2 c | c b | c1 | \bar "||"
}

AltoMusic = \relative {
  e'1 | g2 g | f1 |
  f1 | f2 e | d d | e1 |
}

TenorMusic = \relative {
  c'1 | c2 c | c1 |
  d1 | g,2 g | g g | g1 |
}

BassMusic = \relative {
  c1 | e2 e | f1 |
  d1 | b2 c | g' g | c,1 |
}

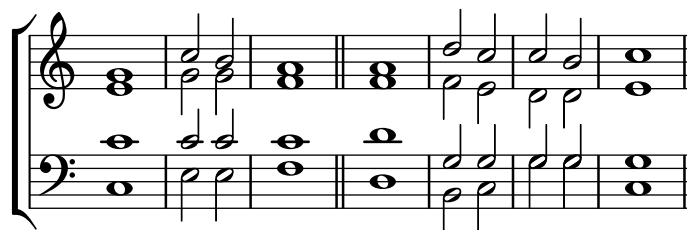
global = {
  \time 2/2
}

% Si usa un blocco markup per centrare il canto sulla pagina
\markup {
  \fill-line {
    \score { % centered
      <<
      \new ChoirStaff <<
        \new Staff <<
          \global
          \clef "treble"
          \new Voice = "Soprano" <<
            \voiceOne
            \SopranoMusic
          >>
          \new Voice = "Alto" <<
            \voiceTwo
            \AltoMusic
          >>
        >>
      \new Staff <<
        \clef "bass"
        \global
        \new Voice = "Tenor" <<
          \voiceOne
          \TenorMusic
        >>
      >>
    }
  }
}
```

```

\new Voice = "Bass" <<
  \voiceTwo
  \BassMusic
>>
>>
>>
  >>
  \layout {
\context {
  \Score
  \override SpacingSpanner.base-shortest-duration = #(ly:make-moment
1/2)
}
\context {
  \Staff
  \remove "Time_signature_engraver"
}
  }
} % End score
}
} % End markup

```



Altri approcci per impostare un canto simile sono illustrati nel primo dei seguenti frammenti.

## Frammenti di codice selezionati

### *Notazione per canti e salmi*

Questa forma di notazione è utilizzata per i salmi, dove i versi non sono sempre della stessa lunghezza.

```

stemOff = \hide Staff.Stem
stemOn  = \undo \stemOff

\score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \key g \minor
    \cadenzaOn
    \stemOff a'\breve bes'4 g'4
    \stemOn a'2 \bar "||"
    \stemOff a'\breve g'4 a'4
    \stemOn f'2 \bar "||"
    \stemOff a'\breve^{\markup { \italic flexe }}
    \stemOn g'2 \bar "||"
  }
}

```



Modello per notazione antica – trascrizione moderna di musica gregoriana

```
\include "gregorian.ly"
```

```
chant = \relative c' {
  \set Score.timing = ##f
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g) a2 \finalis
}

verba = \lyricmode {
  Lo -- rem ip -- sum do -- lor sit a -- met
}

\score {
  \new Staff <<
    \new Voice = "melody" \chant
    \new Lyrics = "one" \lyricsto melody \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Bar_engraver"
      \hide Stem
    }
    \context {
      \Voice
      \override Stem.length = #0
    }
    \context {
      \Score
      barAlways = ##t
    }
  }
}
```



## Vedi anche

Manuale di apprendimento: Sezione “Visibilità e colore degli oggetti” in *Manuale di Apprendimento*, Sezione “Modelli per gruppi vocali” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 2.9 [Ancient notation], pagina 431, [\[Bar lines\]](#), pagina [\[undefined\]](#), Sezione 5.1.4 [Modifying context plug-ins], pagina 587, Sezione 2.9.4 [Typesetting Gregorian chant], pagina 442, [\[undefined\]](#) [Unmetered music], pagina [\[undefined\]](#), Sezione 5.4.7 [Visibility of objects], pagina 618.

## Salmi

Le parole di un salmo anglicano sono solitamente stampate in versi separati centrati sotto il canto.

I canti singoli (7 battute) si ripetono per ogni verso. I canti doppi (con 14 battute) si ripetono per ogni coppia di versi. Sono inseriti dei segni nelle parole per mostrare come si combinano col canto. Ogni verso è diviso a metà e tale divisione è indicata solitamente dai due punti, che corrispondono alla doppia stanghetta in musica. Le parole che precedono i due punti sono cantate insieme alle prime tre battute della musica; quelle successive insieme alle restanti quattro battute.

Stanghetta singola (o in alcuni libri di salmi una virgola inversa o segno simile) sono inserite tra le parole per indicare dove cadono le stanghette nella musica. In modalità markup una stanghetta singola può essere inserita usando il simbolo di controllo battuta |.

```
\markup {
  \fill-line {
    \column {
      \left-align {
        \line { 0 come let us sing | unto the | Lord : let }
        \line { us heartily rejoice in the | strength of | our }
        \line { sal- | -vation. }
      }
    }
  }
}
```

O come let us sing | unto the | Lord : let  
us heartily rejoice in the | strength of | our  
sal- | -vation.

Altri simboli potrebbero richiedere i glifi dei tipi di carattere `fetaMusic`. Maggiori informazioni in [\[undefined\]](#) [Fonts], pagina [\[undefined\]](#).

```
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line { 0 come let us sing \tick unto the \tick Lord : let }
        \line {
          us heartily rejoice in the \tick strength of \tick our
        }
        \line { sal \tick vation. }
      }
    }
  }
}
```



```

    }
  }
}

```

O come let us sing 'unto the 'Lord : let  
us heartily rejoice in the 'strength of 'our  
sal 'vation.

Se c'è una semibreve in una battuta, tutte le parole che si riferiscono a quella battuta sono recitate su quella nota singola col ritmo del parlato. Dove ci sono due note in una battuta ci saranno solo una o due sillabe corrispondenti. Se ci sono più di due sillabe, si inserisce solitamente un punto per indicare dove si trova il cambio di nota.

```

dot = \markup {
  \raise #0.7 \musicglyph #"dots.dot"
}
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line {
          O come let us sing \tick unto \dot the \tick Lord : let
        }
        \line {
          us heartily rejoice in the \tick strength of \tick our
        }
        \line { sal \tick vation. }
      }
    }
  }
}

```

O come let us sing 'unto • the 'Lord : let  
us heartily rejoice in the 'strength of 'our  
sal 'vation.

In alcuni libri di salmi si usa un asterisco, al posto di una virgola, per indicare una pausa in una sezione recitata, mentre le sillabe accentate o leggermente allungate sono indicate in grassetto.

```

dot = \markup {
  \raise #0.7 \musicglyph #"dots.dot"
}
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {

```

```

\line { Today if ye will hear his voice * }
\line {
  \concat { \bold hard en }
  | not your | hearts : as in the pro-
}
\line { vocation * and as in the \bold day of tempt- | }
\line { -ation | in the | wilderness. }
}
}
}
}
}

```

Today if ye will hear his voice \*  
**harden** | not your | hearts : as in the pro-  
vocation \* and as in the **day** of tempt- |  
-ation | in the | wilderness.

In altri libri di salmi si usa un simbolo di accento sopra la sillaba per indicare l'accento.

```

tick = \markup {
  \raise #2 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line {
          O come let us \concat {
            si \combine \tick ng
          }
          | unto the | Lord : let
        }
        \line {
          us heartily \concat {
            rejo \combine \tick ice
          }
          in the | strength of | our
        }
        \line { sal- | -vation. }
      }
    }
  }
}
}
}

```

O come let us síng | unto the | Lord : let  
us heartily rejoyce in the | strength of | our  
sal- | -vation.

L'uso di `\markup` per centrare il testo e disporre le linee in colonne è descritto in [\[Formatting text\]](#), pagina [\[undefined\]](#).

La maggior parte di questi elementi sono mostrati in uno dei due versi del modello Sezione “Salmi” in *Manuale di Apprendimento*.

## Vedi anche

Manuale di apprendimento: Sezione “Salmi” in *Manuale di Apprendimento*, Sezione “Modelli per gruppi vocali” in *Manuale di Apprendimento*.

Guida alla notazione: [\[Fonts\]](#), pagina [\[Formatting text\]](#), pagina [\[Formatting text\]](#).

## Misure parziali nei motivi degli inni

Le melodie degli inni spesso iniziano e terminano ogni rigo musicale con misure parziali, così che ciascun rigo musicale corrisponda esattamente a un rigo di testo. Per fare ciò è necessario un comando `\partial` all’inizio della musica e dei comandi `\bar " | "` o `\bar " | | "` alla fine di ogni linea.

*Modello per inno*

Il codice seguente presenta un modo di impostare un inno in cui ogni verso inizia e finisce con una misura parziale. Mostra anche come aggiungere delle strofe come testo separato sotto la musica.

```
Timeline = {
  \time 4/4
  \tempo 4=96
  \partial 2
  s2 | s1 | s2 \breathes s2 | s1 | s2 \bar " | | " \break
  s2 | s1 | s2 \breathes s2 | s1 | s2 \bar " | | "
}
```

```
SopranoMusic = \relative g' {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}
```

```
AltoMusic = \relative c' {
  d4 d | d d d d | d d d d | d d d d | d2
  d4 d | d d d d | d d d d | d d d d | d2
}
```

```
TenorMusic = \relative a {
  b4 b | b b b b | b b b b | b b b b | b2
  b4 b | b b b b | b b b b | b b b b | b2
}
```

```
BassMusic = \relative g {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}
```

```
global = {
  \key g \major
}
```

```
\score { % Start score
  <<
  \new PianoStaff << % Start pianostaff
  \new Staff << % Start Staff = RH
```

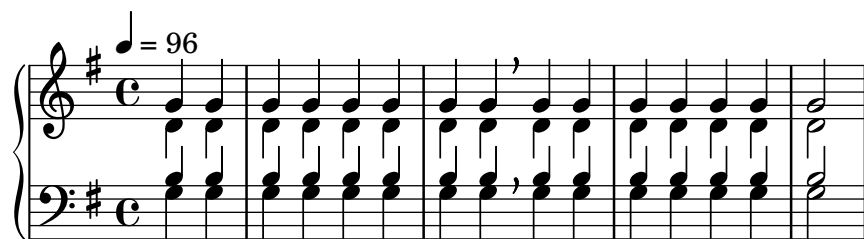
```

\global
\clef "treble"
\new Voice = "Soprano" << % Start Voice = "Soprano"
  \Timeline
  \voiceOne
  \SopranoMusic
>> % End Voice = "Soprano"
\new Voice = "Alto" << % Start Voice = "Alto"
  \Timeline
  \voiceTwo
  \AltoMusic
>> % End Voice = "Alto"
>> % End Staff = RH
\new Staff << % Start Staff = LH
  \global
  \clef "bass"
  \new Voice = "Tenor" << % Start Voice = "Tenor"
    \Timeline
    \voiceOne
    \TenorMusic
  >> % End Voice = "Tenor"
  \new Voice = "Bass" << % Start Voice = "Bass"
    \Timeline
    \voiceTwo
    \BassMusic
  >> % End Voice = "Bass"
>> % End Staff = LH
>> % End pianostaff
>>
} % End score

\markup {
  \fill-line {
    ""
    {
      \column {
        \left-align {
          "This is line one of the first verse"
          "This is line two of the same"
          "And here's line three of the first verse"
          "And the last line of the same"
        }
      }
    }
  }
  ""
}

\paper { % Start paper block
  indent = 0 % don't indent first system
  line-width = 130 % shorten line length to suit music
} % End paper block

```



This is line one of the first verse  
 This is line two of the same  
 And here's line three of the first verse  
 And the last line of the same

### 2.1.8 Musica vocale antica

È supportata la musica vocale antica, come spiegato in Sezione 2.9 [Ancient notation], pagina 431.

#### Vedi anche

Guida alla notazione: Sezione 2.9 [Ancient notation], pagina 431.

## 2.2 Keyboard and other multi-staff instruments

**Un peu retenu**  
*très expressif*

*ppp*

*Rall.*

*long*

*ped.*

*a Tempo*

*pp*



This section discusses several aspects of music notation that are unique to keyboard instruments and other instruments notated on many staves, such as harps and vibraphones. For the purposes of this section this entire group of multi-staff instruments is called “keyboards” for short, even though some of them do not have a keyboard.

### 2.2.1 Common notation for keyboards

This section discusses notation issues that may arise for most keyboard instruments.

#### References for keyboards

Keyboard instruments are usually notated with Piano staves. These are two or more normal staves coupled with a brace. The same notation is also used for other keyed instruments. Organ music is normally written with two staves inside a `PianoStaff` group and third, normal staff for the pedals.

The staves in keyboard music are largely independent, but sometimes voices can cross between the two staves. This section discusses notation techniques particular to keyboard music.

Several common issues in keyboard music are covered elsewhere:

- Keyboard music usually contains multiple voices and the number of voices may change regularly; this is described in [Collision resolution](#), pagina [undefined](#).
- Keyboard music can be written in parallel, as described in [Writing music in parallel](#), pagina [undefined](#).
- Dynamics may be placed in a **Dynamics** context, between the two **Staff** contexts to align the dynamic marks on a horizontal line centered between the staves; see [Dynamics](#), pagina [undefined](#).
- Fingerings are indicated with [Fingering instructions](#), pagina [undefined](#).
- Organ pedal indications are inserted as articulations, see [List of articulations](#), pagina [undefined](#).
- Vertical grid lines can be shown with [Grid lines](#), pagina [undefined](#).
- Keyboard music often contains *Laissez vibrer* ties as well as ties on arpeggios and tremolos, described in [Ties](#), pagina [undefined](#).
- Placing arpeggios across multiple voices and staves is covered in [Arpeggio](#), pagina 144.
- Tremolo marks are described in [Tremolo repeats](#), pagina [undefined](#).
- Several of the tweaks that can occur in keyboard music are demonstrated in Sezione “Real music example” in *Manuale di Apprendimento*.
- Hidden notes can be used to produce ties that cross voices, as shown in Sezione “Other uses for tweaks” in *Manuale di Apprendimento*.

## Vedi anche

Learning Manual: Sezione “Real music example” in *Manuale di Apprendimento*, Sezione “Other uses for tweaks” in *Manuale di Apprendimento*.

Notation Reference: [\[Grouping staves\]](#), pagina [\[Instrument names\]](#), pagina [\[Collision resolution\]](#), pagina [\[Writing music in parallel\]](#), pagina [\[Fingering instructions\]](#), pagina [\[List of articulations\]](#), pagina [\[Grid lines\]](#), pagina [\[Ties\]](#), pagina [\[Arpeggio\]](#), pagina 144, [\[Tremolo repeats\]](#), pagina [\[Ties\]](#).

Internals Reference: Sezione “PianoStaff” in *Guida al Funzionamento Interno*.

Snippets: Sezione “Keyboards” in *Frammenti di codice*.

## Changing staff manually

Voices can be switched between staves manually, using the command

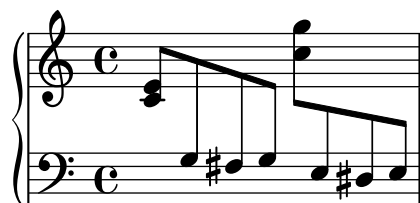
```
\change Staff = staffname
```

The string *staffname* is the name of the staff. It switches the current voice from its current staff to the staff called *staffname*. Typical values for *staffname* are "up" and "down", or "RH" and "LH".

The staff to which the voice is being switched must exist at the time of the switch. If necessary, staves should be “kept alive”, see Sezione 5.1.3 [Keeping contexts alive], pagina 584.

Cross-staff notes are beamed automatically:

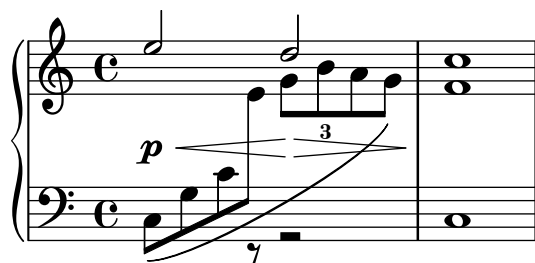
```
\new PianoStaff <<
  \new Staff = "up" {
    <e' c'>8
    \change Staff = "down"
    g8 fis g
    \change Staff = "up"
    <g' ' c''>8
    \change Staff = "down"
    e8 dis e
    \change Staff = "up"
  }
  \new Staff = "down" {
    \clef bass
    % keep staff alive
    s1
  }
>>
```



If the beaming needs to be tweaked, make any changes to the stem directions first. The beam positions are then measured from the center of the staff that is closest to the beam. For a simple example of beam tweaking, see Sezione “Fixing overlapping notation” in *Manuale di Apprendimento*.

Overlapping notation can result when voices cross staves:

```
\new PianoStaff <<
  \new Staff = "up" {
    \voiceOne
    % Make space for fingering in the cross-staff voice
    \once\override DynamicLineSpanner.staff-padding = #4
    e''2\p\< d''\>
    c''1\!
  }
  \new Staff = "down" <<
  {
    \clef bass
    s4. e,8\rest g,2\rest
    c1
  } \ {
    c8\< g c'
    \change Staff = "up"
    e' g' b'-3 a' g'\>
    f'1
  }
>>
>>
```



The stem and slur overlap the intervening line of dynamics because automatic collision resolution is suspended for beams, slurs and other spanners that connect notes on different staves, as well as for stems and articulations if their placement is affected by a cross-staff spanner. The resulting collisions must be resolved manually, where necessary, using the methods in Sezione “Fixing overlapping notation” in *Manuale di Apprendimento*.

## Vedi anche

Learning Manual: Sezione “Fixing overlapping notation” in *Manuale di Apprendimento*.

Notation Reference: [\[Stems\]](#), pagina [\[undefined\]](#), [\[Automatic beams\]](#), pagina [\[undefined\]](#), Sezione 5.1.3 [\[Keeping contexts alive\]](#), pagina 584.

Snippets: Sezione “Keyboards” in *Frammenti di codice*.

Internals Reference: Sezione “Beam” in *Guida al Funzionamento Interno*, Sezione “ContextChange” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Beam collision avoidance does not work for automatic beams that end right before a change in staff. In this case use manual beams.



## Changing staff automatically

Voices can be made to switch automatically between the top and the bottom staff. The syntax for this is

```
\autochange ...music...
```

This will create two staves inside the current staff group (usually a `PianoStaff`), called "up" and "down". The lower staff will be in the bass clef by default. The autochanger switches on the basis of the pitch (middle C is the turning point), and it looks ahead skipping over rests to switch in advance.

```
\new PianoStaff {
  \autochange {
    g4 a b c'
    d'4 r a g
  }
}
```

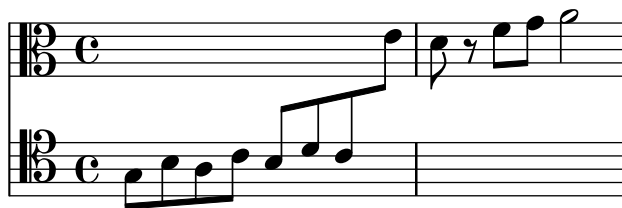


It is possible to specify other pitches for the turning point. If the staves are not instantiated explicitly, other clefs may be used.

```
music = {
  g8 b a c' b8 d' c'8 e'
  d'8 r f' g' a'2
}
```

```
\autochange d' \music
\autochange b \with { \clef soprano } \music
\autochange d' \with { \clef alto } \with { \clef tenor } \music
```





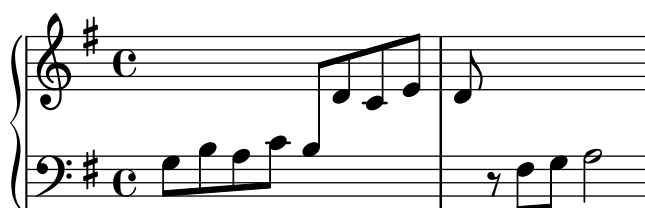
A `\relative` section that is outside of `\autochange` has no effect on the pitches of the music, so if necessary, put `\relative` inside `\autochange`.

If additional control is needed over the individual staves, they can be created manually with the names "up" and "down". The `\autochange` command will then switch its voice between the existing staves.

**Nota:** If staves are created manually, they *must* be named "up" and "down".

For example, staves must be created manually in order to place a key signature in the lower staff:

```
\new PianoStaff <<
  \new Staff = "up" {
    \new Voice = "melOne" {
      \key g \major
      \autochange \relative {
        g8 b a c b d c e
        d8 r fis, g a2
      }
    }
  }
  \new Staff = "down" {
    \key g \major
    \clef bass
  }
>>
```



## Vedi anche

Notation Reference: [Changing staff manually], pagina 326.

Snippets: Sezione "Keyboards" in *Frammenti di codice*.

Internals Reference: Sezione "AutoChangeMusic" in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

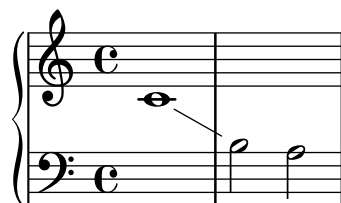
The staff switches may not end up in optimal places. For high quality output, staff switches should be specified manually.

Chords will not be split across the staves; they will be assigned to a staff based on the first note named in the chord construct.

## Staff-change lines

Whenever a voice switches to another staff, a line connecting the notes can be printed automatically:

```
\new PianoStaff <<
  \new Staff = "one" {
    \showStaffSwitch
    c'1
    \change Staff = "two"
    b2 a
  }
  \new Staff = "two" {
    \clef bass
    s1*2
  }
>>
```



## Comandi predefiniti

`\showStaffSwitch`, `\hideStaffSwitch`.

## Vedi anche

Snippets: Sezione “Keyboards” in *Frammenti di codice*.

Internals Reference: Sezione “Note\_head\_line\_engraver” in *Guida al Funzionamento Interno*, Sezione “VoiceFollower” in *Guida al Funzionamento Interno*.

## Cross-staff stems

Chords that cross staves may be produced using the `Span_stem_engraver`. Care must be taken to ensure that automatic beams do not beam the notes on one staff when it’s not required on the other.

```
\layout {
  \context {
    \PianoStaff
    \consists #Span_stem_engraver
  }
}

{
  \new PianoStaff <<
    \new Staff {
      <b d'>4 r d'16\> e'8. g8 r\!
      e'8 f' g'4 e'2
    }
    \new Staff {
      \clef bass
    }
  >>
```

```

\voiceOne
\autoBeamOff
\crossStaff { <e g>4 e, g16 a8. c8} d
\autoBeamOn
g8 f g4 c2
}
>>
}

```



For the time being, this engraver can not be specified by its name in double quotes, but rather prefixing its name with a hash symbol #, due to the way it is implemented.

## Frammenti di codice selezionati

### *Indicating cross-staff chords with arpeggio bracket*

An arpeggio bracket can indicate that notes on two different staves are to be played with the same hand. In order to do this, the `PianoStaff` must be set to accept cross-staff arpeggios and the arpeggios must be set to the bracket shape in the `PianoStaff` context.

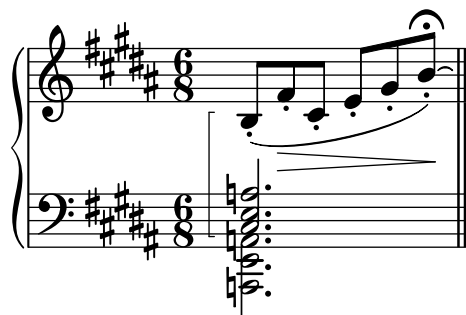
(Debussy, Les collines d'Anacapri, m. 65)

```

\new PianoStaff <<
\set PianoStaff.connectArpeggios = ##t
\override PianoStaff.Arpeggio.stencil = #ly:arpeggio::brew-chord-bracket
\new Staff {
\relative c' {
\key b \major
\time 6/8
b8-.(\arpeggio fis'-.\> cis-. e-. gis-. b-.)\!\fermata^\laissezVibrer
\bar "||"
}
}
\new Staff {
\relative c' {
\clef bass
\key b \major
<<
{
<a e cis>2.\arpeggio
}
\\
{
<a, e a,>2.
}
}
>>
}

```

```
}
>>
```



## Vedi anche

Snippets: Sezione “Keyboards” in *Frammenti di codice*.

Internals Reference: Sezione “Stem” in *Guida al Funzionamento Interno*.

### 2.2.2 Piano

This section discusses notation issues that relate most directly to the piano.

#### Piano pedals

Pianos generally have three pedals that alter the way sound is produced: *sustain*, *sostenuto* (*sos.*), and *una corda* (*U.C.*). Sustain pedals are also found on vibraphones and celestas.

```
\relative {
  c''4\sustainOn d e g
  <c, f a>1\sustainOff
  c4\sostenutoOn e g c,
  <bes d f>1\sostenutoOff
  c4\unaCorda d e g
  <d fis a>1\treCorde
}
```



There are three styles of pedal indications: text, bracket, and mixed. The sustain pedal and the una corda pedal use the text style by default while the sostenuto pedal uses mixed by default.

```
\relative {
  c''4\sustainOn g c2\sustainOff
  \set Staff.pedalSustainStyle = #'mixed
  c4\sustainOn g c d
  d\sustainOff\sustainOn g, c2\sustainOff
  \set Staff.pedalSustainStyle = #'bracket
  c4\sustainOn g c d
  d\sustainOff\sustainOn g, c2
  \bar "|."
}
```



The placement of the pedal commands matches the physical movement of the sustain pedal during piano performance. Pedalling to the final bar line is indicated by omitting the final pedal off command.

Pedal indications may be placed in a `Dynamics` context, which aligns them on a horizontal line.

## Vedi anche

Notation Reference: `<undefined>` [Ties], pagina `<undefined>`.

Snippets: Sezione “Keyboards” in *Frammenti di codice*.

Internals Reference: Sezione “SustainPedal” in *Guida al Funzionamento Interno*, Sezione “SustainPedalLineSpanner” in *Guida al Funzionamento Interno*, Sezione “SustainEvent” in *Guida al Funzionamento Interno*, Sezione “SostenutoPedal” in *Guida al Funzionamento Interno*, Sezione “SostenutoPedalLineSpanner” in *Guida al Funzionamento Interno*, Sezione “SostenutoEvent” in *Guida al Funzionamento Interno*, Sezione “UnaCordaPedal” in *Guida al Funzionamento Interno*, Sezione “UnaCordaPedalLineSpanner” in *Guida al Funzionamento Interno*, Sezione “UnaCordaEvent” in *Guida al Funzionamento Interno*, Sezione “PianoPedalBracket” in *Guida al Funzionamento Interno*, Sezione “Piano-pedal-engraver” in *Guida al Funzionamento Interno*.

## 2.2.3 Accordion

This section discusses notation that is unique to the accordion.

### Discant symbols

Accordions are often built with more than one set of reeds that may be in unison with, an octave above, or an octave below the written pitch. Each accordion maker has different names for the *shifts* that select the various reed combinations, such as *oboe*, *musette*, or *bandonium*, so a system of symbols has come into use to simplify the performance instructions.

## Frammenti di codice selezionati

### *Accordion register symbols*

Accordion register symbols are available as `\markup` as well as as standalone music events (as register changes tend to occur between actual music events. Bass registers are not overly standardized. The available commands can be found in ‘Accordion Registers’ in the Notation Reference.

```
#(use-modules (scm accreg))
```

```
\new PianoStaff
<<
  \new Staff \relative {
    \clef treble \discant "10" r8 s32 f'[ bes f] s e[ a e] s d[ g d] s16 e32[ a]
    <<
      { r16 <f bes> r <e a> r <d g> }
      \\
      { d r a r bes r }
    >> |
    <cis e a>1
  }
  \new Staff \relative {
    \clef treble \freeBass "1" r8 d'32 s16. c32 s16. bes32 s16. a32[ cis] s16
```





or pedal diagrams:

`\textLengthOn`

`cis''1_\markup { \harp-pedal #"^v-|vv-^" }`

`c''!1_\markup { \harp-pedal #"^o--|vv-^" }`



The `\harp-pedal` command accepts a string of characters, where `^` is the highest pedal position (flattened pitch), `-` is the middle pedal position (natural pitch), `v` is the lowest pedal position (sharpened pitch), and `|` is the divider. A prefixed `o` will circle the following pedal symbol.

## Vedi anche

Notation Reference: [\[Text scripts\]](#), pagina [\[undefined\]](#), Sezione A.11.5 [\[Instrument Specific Markup\]](#), pagina 722.

## 2.3 Unfretted string instruments

**lentement**

1 *fatigué* s. vib. n. p. vib. s. vib.

IV IV IV

*mf* *mf* *mf* *ff* *pp*

**accel...** s.p. n. s.p. n. p. vib.

IV IV

*mf* *ff*

s.p. n. s.p. n. p. vib. m. vib.

*ritar...*

IV IV IV

*ppp*

This section provides information and references which are helpful when writing for unfretted string instruments, principally orchestral strings.



### 2.3.1 Common notation for unfretted strings

There is little specialist notation for unfretted string instruments. The music is notated on a single staff, and usually only a single voice is required. Two voices might be required for some double-stopped or divisi passages.

#### References for unfretted strings

Most of the notation which is useful for orchestral strings and other bowed instruments is covered elsewhere:

- Textual indications such as “pizz.” and “arco” are added as simple text – see [\[Text scripts\]](#), pagina [\[undefined\]](#).
- Fingerings, including the thumb indication, are described in [\[Fingering instructions\]](#), pagina [\[undefined\]](#).
- Double stopping is normally indicated by writing a chord, see [\[Chorded notes\]](#), pagina [\[undefined\]](#). Directives for playing chords may be added, see [\[Arpeggio\]](#), pagina 144.
- Templates for string quartets can be found in Sezione “String quartet templates” in *Manuale di Apprendimento*. Others are shown in the snippets.

#### Vedi anche

Learning Manual: Sezione “String quartet templates” in *Manuale di Apprendimento*.

Notation Reference: [\[Text scripts\]](#), pagina [\[undefined\]](#), [\[Fingering instructions\]](#), pagina [\[undefined\]](#), [\[Chorded notes\]](#), pagina [\[undefined\]](#), [\[Arpeggio\]](#), pagina 144.

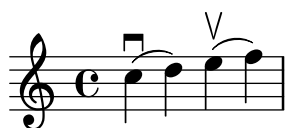
Snippets: Sezione “Unfretted strings” in *Frammenti di codice*.

#### Bowing indications

Bowing indications are created as articulations, which are described in [\[Articulations and ornamentations\]](#), pagina [\[undefined\]](#).

The bowing commands, `\upbow` and `\downbow`, are used with slurs as follows:

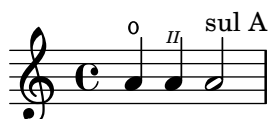
```
\relative { c'4(\downbow d) e(\upbow f) }
```



Roman numerals can be added as strings numbers (rather than the default circled Arabic numbers), as explained in [\[String number indications\]](#), pagina 339.

Alternatively, string indications may be printed using markup commands; articulation scripts may also indicate open strings.

```
a'4 \open
\romanStringNumbers
a'\2
a'2^\markup { \small "sul A" }
```



## Comandi predefiniti

`\downbow`, `\upbow`, `\open`, `\romanStringNumbers`.

## Vedi anche

Notation Reference: [\[Articulations and ornamentations\]](#), pagina [\[undefined\]](#), [\[String number indications\]](#), pagina [339](#), [\[undefined\]](#) [\[Slurs\]](#), pagina [\[undefined\]](#).

## Harmonics

### *Natural harmonics*

Natural harmonics can be notated in several ways. A diamond-shaped note head generally means to touch the string where you would stop the note if it were not a diamond.

```
\relative d'' {
  d4 e4.
  \harmonicsOn
  d8 e e
  d4 e4.
  \harmonicsOff
  d8 e e
}
```



Alternatively a normal note head is shown at the pitch to be sounded together with a small circle to indicate it should be played as a harmonic:

```
d''2^\flageolet d''_\flageolet
```



A smaller circle may be created, see the snippet list in [\[References for unfretted strings\]](#), pagina [336](#).

### *Artificial harmonics*

Artificial harmonics are notated with two notes, one with a normal note head indicating the stopped position and one with an open diamond note head to indicate the harmonic position.

Artificial harmonics indicated with `\harmonic` do not show the dots. The context property `harmonicDots` should be set if dots are required.

```
\relative e' {
  <e a\harmonic>2. <c g'\harmonic>4
  \set harmonicDots = ##t
  <e a\harmonic>2. <c g'\harmonic>4
}
```



**Nota:** `\harmonic must` be placed inside a chord construct even if there is only a single note. Normally `\harmonicsOn` would be used in this situation.

## Vedi anche

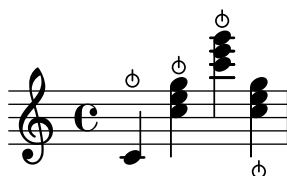
Music Glossary: Sezione “harmonics” in *Glossario Musicale*.

Notation Reference: `<undefined>` [Special note heads], pagina `<undefined>`, [References for unfretted strings], pagina 336.

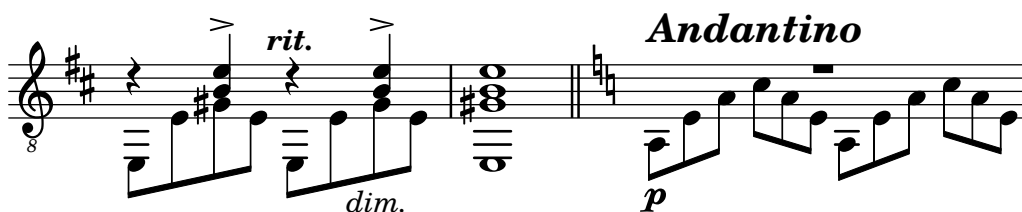
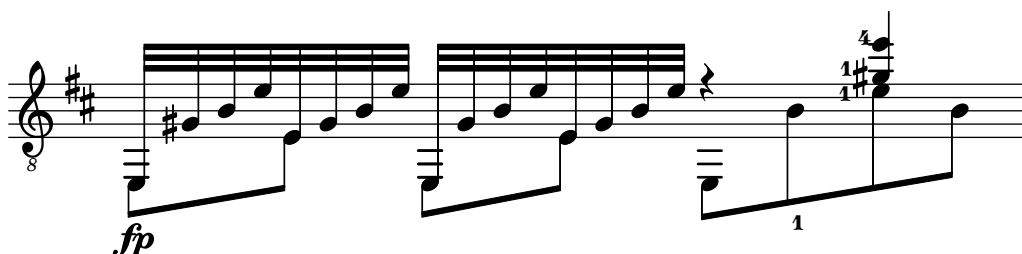
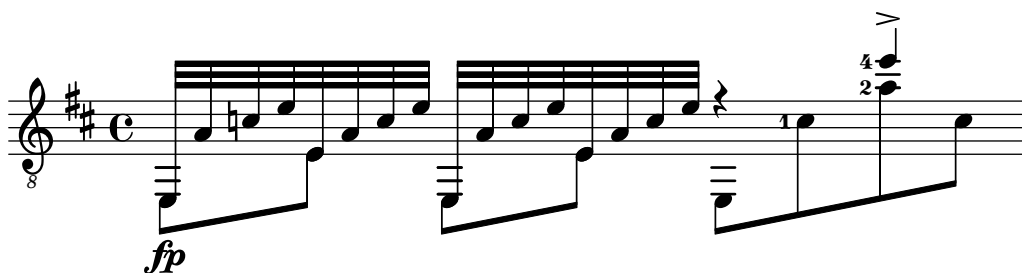
## Snap (Bartók) pizzicato

A *snap pizzicato* (also known as “Bartok pizz”) is a type of pizzicato where the string is deliberately plucked upwards (rather than sideways) such that it hits the fingerboard.

```
\relative {
  c'4\snappizzicato
  <c' e g>4\snappizzicato
  <c' e g>4^\snappizzicato
  <c, e g>4_\snappizzicato
}
```



## 2.4 Fretted string instruments



*Andantino*

*p*



This section discusses several aspects of music notation that are unique to fretted string instruments.

### 2.4.1 Common notation for fretted strings

This section discusses common notation that is unique to fretted string instruments.

#### References for fretted strings

Music for fretted string instruments is normally notated on a single staff, either in traditional music notation or in tablature. Sometimes the two types are combined, and it is especially common in popular music to use chord diagrams above a staff of traditional notation. The guitar and the banjo are transposing instruments, sounding an octave lower than written. Scores for these instruments should use the "treble\_8" clef (or `\transposition c` to get correct MIDI output). Some other elements pertinent to fretted string instruments are covered elsewhere:

- Fingerings are indicated as shown in [\[Fingering instructions\]](#), pagina [\[undefined\]](#).
- Instructions for *Laissez vibrer* ties as well as ties on arpeggios and tremolos can be found in [\[Ties\]](#), pagina [\[undefined\]](#).
- Instructions for handling multiple voices can be found in [\[Collision resolution\]](#), pagina [\[undefined\]](#).
- Instructions for indicating harmonics can be found in [\[Harmonics\]](#), pagina 337.

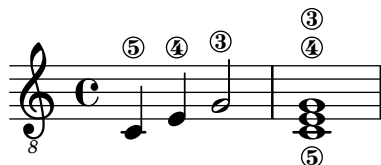
#### Vedi anche

Notation Reference: [\[Fingering instructions\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Ties\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Collision resolution\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Instrument names\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Writing music in parallel\]](#), pagina [\[undefined\]](#), [\[Arpeggio\]](#), pagina 144, [\[undefined\]](#) [\[List of articulations\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Clef\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Instrument transpositions\]](#), pagina [\[undefined\]](#).

#### String number indications

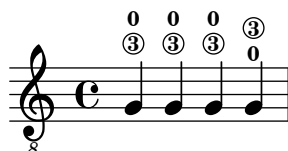
The string on which a note should be played may be indicated by appending `\number` to a note. `\clef "treble_8"`

```
c4\5 e\4 g2\3
<c\5 e\4 g\3>1
```



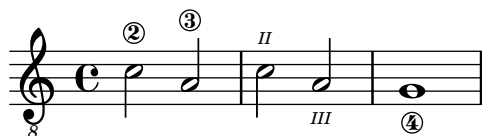
When fingerings and string indications are used together, their placement can be controlled by the order in which the two items appear in the code *only* if they appear inside of an explicit chord: applied to whole chords or single notes *outside* of chords, fingerings are placed using a different mechanism.

```
\clef "treble_8"
g4\3-0
g-0\3
<g\3-0>
<g-0\3>
```



String numbers may also, as is customary with unfretted strings, be printed in Roman numerals and placed below the staff rather than above.

```
\clef "treble_8"
c'2\2
a\3
\romanStringNumbers
c'\2
\set stringNumberOrientations = #'(down)
a\3
\arabicStringNumbers
g1\4
```



## Frammenti di codice selezionati

*Controllare il posizionamento delle diteggiature di un accordo*

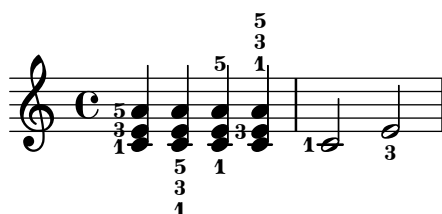
Il posizionamento dei numeri della diteggiatura può essere regolato in modo preciso. Perché l'orientamento funzioni, occorre usare il costrutto per gli accordi `<>` anche per una nota singola.

```
\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down)
  <c-1 e-3 a-5>4
```

```

\set fingeringOrientations = #'(down right up)
<c-1 e-3 a-5>4
\set fingeringOrientations = #'(up)
<c-1 e-3 a-5>4
\set fingeringOrientations = #'(left)
<c-1>2
\set fingeringOrientations = #'(down)
<e-3>2
}

```



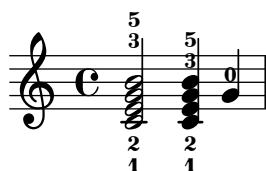
*Far sì che la diteggiatura appaia dentro il rigo*

Per impostazione predefinita, le diteggiature orientate verticalmente sono poste fuori dal rigo. Tuttavia, questo comportamento può essere annullato. Attenzione: bisogna usare il costrutto per gli accordi `<>`, anche se si riferisce a una singola nota.

```

\relative c' {
  <c-1 e-2 g-3 b-5>2
  \override Fingering.staff-padding = #'()
  <c-1 e-2 g-3 b-5>4 <g'-0>
}

```



## Comandi predefiniti

`\arabicStringNumbers`, `\romanStringNumbers`.

## Vedi anche

Notation Reference: `<undefined>` [Fingering instructions], pagina `<undefined>`.

Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

Internals Reference: Sezione “StringNumber” in *Guida al Funzionamento Interno*, Sezione “Fingering” in *Guida al Funzionamento Interno*.

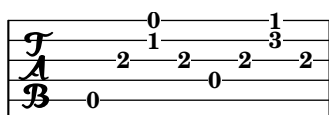
## Default tablatures

Music for plucked string instruments is frequently notated using a finger/touch notation or tablature. In contrast to traditional notation pitches are not denoted with note heads, but by numbers (or letter-like symbols in historical intavolatura). The staff lines in tablature indicate the string on which the note is to be played, and a number placed on a staff line indicated the fret at which the corresponding string is to be pressed. Notes that are to be played simultaneously are vertically aligned.

By default, string 1 is the highest string, and corresponds to the top line on the `TabStaff`. The tuning of the `TabStaff` strings defaults to the standard guitar tuning (with 6 strings).

The notes are printed as tablature, by using `TabStaff` and `TabVoice` contexts. A calligraphic tablature clef is added automatically.

```
\new TabStaff \relative {
  a,8 a' <c e> a
  d,8 a' <d f> a
}
```



Default tablatures do not contain any symbols for tone duration nor any other musical symbols such as e.g. expressive marks.

```
symbols = {
  \time 3/4
  c4-.^"Allegro" d( e)
  f4-. \f g a^ \fermata
  \mark \default
  c8_. \<\( c16 c~ 2\!
  c'2. \prall\}
}
```

```
\score {
  <<
    \new Staff { \clef "G_8" \symbols }
    \new TabStaff { \symbols }
  >>
}
```

If all musical symbols used in traditional notation should also show up in tablature one has to apply the command `\tabFullNotation` in a `TabStaff`-context. Please bear in mind that half notes are double-stemmed in tablature in order to distinguish them from quarter notes.

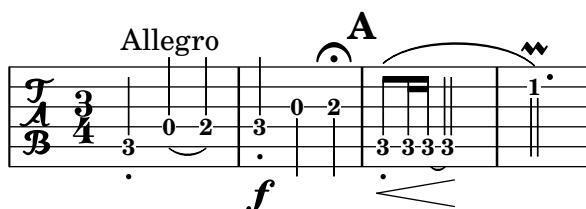
```
symbols = {
  \time 3/4
  c4-.^"Allegro" d( e)
  f4-. \f g a^ \fermata
  \mark \default
  c8_. \<\( c16 c~ 2\!
  c'2. \prall\}
}
```

```
\score {
```

```

\new TabStaff {
  \tabFullNotation
  \symbols
}

```



By default pitches are assigned to the lowest playing position on the fret-board (first position). Open strings are automatically preferred. If you would like a certain pitch to be played on a specific string you can add a string number indication to the pitch name. If you don't want to have string number indications appear in traditional notation, you can override the respective stencil. Usually it will be more comfortable to define the playing position by using the value of `minimumFret`. The default value for `minimumFret` is 0.

Even when `minimumFret` is set, open strings are used whenever possible. This behaviour can be changed by setting `restrainOpenStrings` to `#t`.

```

\layout { \omit Voice.StringNumber }
\new StaffGroup <<
  \new Staff \relative {
    \clef "treble_8"
    \time 2/4
    c16 d e f g4
    c,16\5 d\5 e\4 f\4 g4\4
    c,16 d e f g4
  }
  \new TabStaff \relative {
    c16 d e f g4
    c,16\5 d\5 e\4 f\4 g4\4
    \set TabStaff.minimumFret = #5
    \set TabStaff.restrainOpenStrings = ##t
    c,16 d e f g4
  }
>>

```

Chord constructs can be repeated by the chord repetition symbol `q`. In combination with tabulatures, its behavior of removing string and finger numbers alongside with other events is cumbersome, so you'll want to run

```

\chordRepeats #'(string-number-event fingering-event)

```



explicitly on music expressions in tabulature using `\repeat` [Chord repetition], pagina `\repeat`. This particular command is so common that it is available as `\tabChordRepeats`.

```
guitar = \relative {
  r8 <gis-2 cis-3 b-0>~ q4 q8~ 8 q4
}
```

```
\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \guitar
  }
  \new TabStaff {
    \tabChordRepeats \guitar
  }
>>
```

Ties over a line break are parenthesized by default. The same holds for the second alternative of a repeat.

```
ties = \relative {
  \repeat volta 2 {
    e'2. f4~
    2 g2~
  }
  \alternative {
    { g4 f2. }
    { g4\repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}
```

```
\score {
  <<
    \new StaffGroup <<
      \new Staff {
        \clef "treble_8"
        \ties
      }
      \new TabStaff {
        \ties
      }
    >>
  >>
}
```

```

>>
\layout {
  indent = #0
  ragged-right = ##t
}
}

```

The command `\hideSplitTiedTabNotes` cancels the behavior of engraving fret numbers in parentheses:

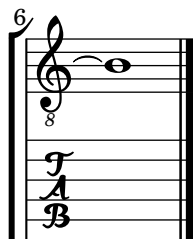
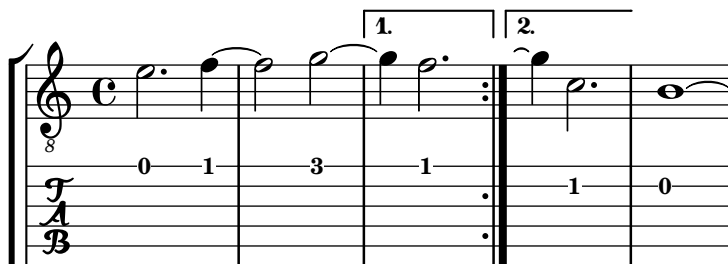
```

ties = \relative {
  \repeat volta 2 {
    e'2. f4~
    2 g2~ }
  \alternative {
    { g4 f2. }
    { g4\repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}

\score {
  <<
  \new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \ties
  }
  \new TabStaff {
    \hideSplitTiedTabNotes
    \ties
  }
  >>
}

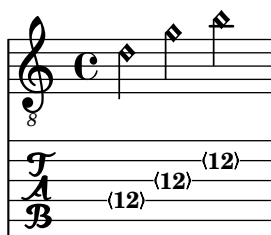
```

```
>>
\layout {
  indent = #0
  ragged-right = ##t
}
}
```



Harmonic indications can be added to tablature notation as sounding pitches:

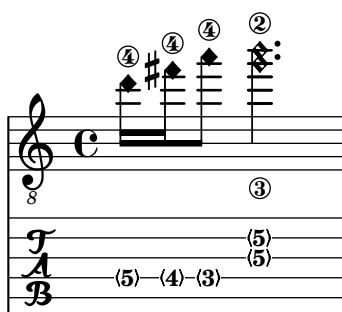
```
\layout { \omit Voice.StringNumber }
firstHarmonic = {
  d'4\4\harmonic
  g'4\3\harmonic
  b'2\2\harmonic
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \firstHarmonic
    }
    \new TabStaff { \firstHarmonic }
  >>
}
```



Note that the command `\harmonic` must always be attached to single notes (possibly inside of a chord) instead of whole chords. It only makes sense for open-string harmonics in the 12th

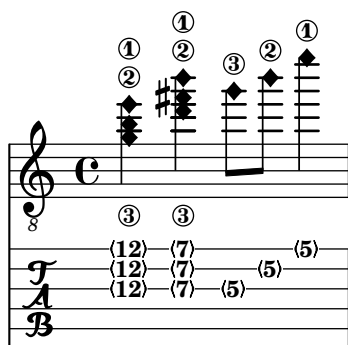
fret. All other harmonics should be calculated by LilyPond. This can be achieved by indicating the fret where a finger of the fretting hand should touch a string.

```
fretHarmonics = {
  \harmonicByFret #5 d16\4
  \harmonicByFret #4 d16\4
  \harmonicByFret #3 d8\4
  \harmonicByFret #5 <g\3 b\2>2.
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \fretHarmonics
    }
    \new TabStaff { \fretHarmonics }
  >>
}
```



Alternatively, harmonics can be computed by defining the ratio of string lengths above and below the harmonic fingering.

```
ratioHarmonics = {
  \harmonicByRatio #1/2 <g\3 b\2 e'\1>4
  \harmonicByRatio #1/3 <g\3 b\2 e'\1>4
  \harmonicByRatio #1/4 { g8\3 b8\2 e'4\1 }
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \ratioHarmonics
    }
    \new TabStaff { \ratioHarmonics }
  >>
}
```

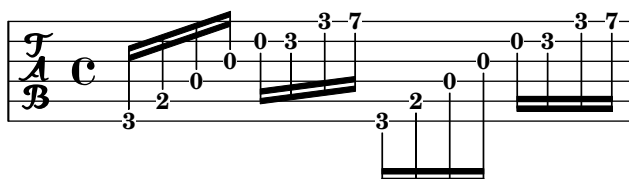


## Frammenti di codice selezionati

### *Stem and beam behavior in tablature*

The direction of stems is controlled the same way in tablature as in traditional notation. Beams can be made horizontal, as shown in this example.

```
\new TabStaff {
  \relative c {
    \tabFullNotation
    g16 b d g b d g b
    \stemDown
    \override Beam.concaveness = #10000
    g,,16 b d g b d g b
  }
}
```



### *Polyphony in tablature*

Polyphony is created the same way in a TabStaff as in a regular staff.

```
upper = \relative c' {
  \time 12/8
  \key e \minor
  \voiceOne
  r4. r8 e, fis g16 b g e e' b c b a g fis e
}
```

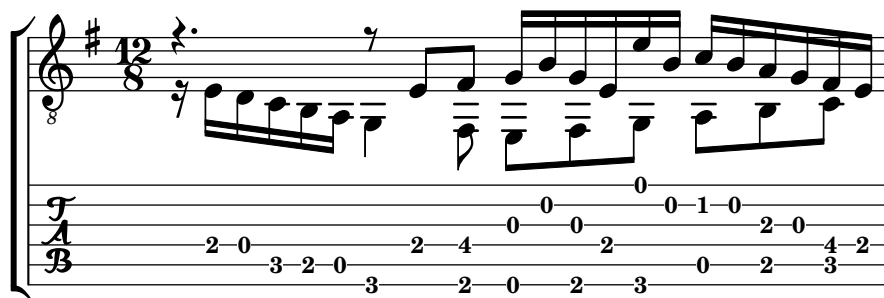
```
lower = \relative c {
  \key e \minor
  \voiceTwo
  r16 e d c b a g4 fis8 e fis g a b c
}
```

```
\score {
  <<
    \new StaffGroup = "tab with traditional" <<
      \new Staff = "guitar traditional" <<
        \clef "treble_8"
        \context Voice = "upper" \upper
        \context Voice = "lower" \lower
```

```

>>
\new TabStaff = "guitar tab" <<
  \context TabVoice = "upper" \upper
  \context TabVoice = "lower" \lower
>>
>>
>>
}

```



*Open string harmonics in tablature*

This snippet demonstrates open-string harmonics

```

openStringHarmonics = {
  \textSpannerDown
  \override TextSpanner.staff-padding = #3
  \override TextSpanner.dash-fraction = #0.3
  \override TextSpanner.dash-period = #1

  %first harmonic
  \override TextSpanner.bound-details.left.text = \markup\small "1st harm. "
  \harmonicByFret #12 e,\6\startTextSpan
  \harmonicByRatio #1/2 e,\6\stopTextSpan

  %second harmonic
  \override TextSpanner.bound-details.left.text = \markup\small "2nd harm. "
  \harmonicByFret #7 e,\6\startTextSpan
  \harmonicByRatio #1/3 e,\6
  \harmonicByFret #19 e,\6
  \harmonicByRatio #2/3 e,\6\stopTextSpan
  %\harmonicByFret #19 < e,\6 a,\5 d\4 >
  %\harmonicByRatio #2/3 < e,\6 a,\5 d\4 >

  %third harmonic
  \override TextSpanner.bound-details.left.text = \markup\small "3rd harm. "
  \harmonicByFret #5 e,\6\startTextSpan
  \harmonicByRatio #1/4 e,\6
  \harmonicByFret #24 e,\6
  \harmonicByRatio #3/4 e,\6\stopTextSpan
  \break

  %fourth harmonic
  \override TextSpanner.bound-details.left.text = \markup\small "4th harm. "
  \harmonicByFret #4 e,\6\startTextSpan

```

```

\harmonicByRatio #1/5 e,\6
\harmonicByFret #9 e,\6
\harmonicByRatio #2/5 e,\6
\harmonicByFret #16 e,\6
\harmonicByRatio #3/5 e,\6\stopTextSpan

% fifth harmonic
\override TextSpanner.bound-details.left.text = \markup\small "5th harm. "
\harmonicByFret #3 e,\6\startTextSpan
\harmonicByRatio #1/6 e,\6\stopTextSpan
\break

%sixth harmonic
\override TextSpanner.bound-details.left.text = \markup\small "6th harm. "
\harmonicByFret #2.7 e,\6\startTextSpan
\harmonicByRatio #1/7 e,\6\stopTextSpan

%seventh harmonic
\override TextSpanner.bound-details.left.text = \markup\small "7th harm. "
\harmonicByFret #2.3 e,\6\startTextSpan
\harmonicByRatio #1/8 e,\6\stopTextSpan

%eighth harmonic
\override TextSpanner.bound-details.left.text = \markup\small "8th harm. "
\harmonicByFret #2 e,\6\startTextSpan
\harmonicByRatio #1/9 e,\6\stopTextSpan
}

\score {
  <<
    \new Staff
    \with { \omit StringNumber } {
      \new Voice {
        \clef "treble_8"
        \openStringHarmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \openStringHarmonics
      }
    }
  >>
}

```

1st harm.      2nd harm.      3rd harm.

(12) (12)      (7) (7)      (19) (19)      (5) (5)      (24) (24)

6

8 4th harm. .... 5th harm.

10 6th harm. .... 7th harm. .... 8th harm. ....

(4) (4) (9) (9) (16) (16) (3) (3)

(2.7) (2.7) (2.3) (2.3) (2) (2)

### *Fretted-string harmonics in tablature*

Demonstrates fretted-string harmonics in tablature

```
pinchedHarmonics = {
  \textSpannerDown
  \override TextSpanner.bound-details.left.text =
    \markup { \halign #-0.5 \teeny "PH" }
  \override TextSpanner.style =
    #'dashed-line
  \override TextSpanner.dash-period = #0.6
  \override TextSpanner.bound-details.right.attach-dir = #1
  \override TextSpanner.bound-details.right.text =
    \markup { \draw-line #'(0 . 1) }
  \override TextSpanner.bound-details.right.padding = #-0.5
}

harmonics = {
  %artificial harmonics (AH)
  \textLengthOn
  <\parenthesize b b'\harmonic>4\_markup{ \teeny "AH 16" }
  <\parenthesize g g'\harmonic>4\_markup{ \teeny "AH 17" }
  <\parenthesize d' d'\harmonic>2\_markup{ \teeny "AH 19" }
  %pinched harmonics (PH)
  \pinchedHarmonics
  <a'\harmonic>2\startTextSpan
  <d'\harmonic>4
  <e'\harmonic>4\stopTextSpan
  %tapped harmonics (TH)
  <\parenthesize g\4 g'\harmonic>4\_markup{ \teeny "TH 17" }
  <\parenthesize a\4 a'\harmonic>4\_markup{ \teeny "TH 19" }
  <\parenthesize c'\3 c'\harmonic>2\_markup{ \teeny "TH 17" }
  %touch harmonics (TCH)
  a4( <e'\harmonic>2. )\_markup{ \teeny "TCH" }
}
```

```
frettedStrings = {
  %artificial harmonics (AH)
```



```

\harmonicByFret #4 g4\3
\harmonicByFret #5 d4\4
\harmonicByFret #7 g2\3
%pinched harmonics (PH)
\harmonicByFret #7 d2\4
\harmonicByFret #5 d4\4
\harmonicByFret #7 a4\5
%tapped harmonics (TH)
\harmonicByFret #5 d4\4
\harmonicByFret #7 d4\4
\harmonicByFret #5 g2\3
%touch harmonics (TCH)
a4 \harmonicByFret #9 g2.\3
}

\score {
  <<
    \new Staff
    \with { \omit StringNumber } {
      \new Voice {
        \clef "treble_8"
        \harmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \frettedStrings
      }
    }
  >>
}

```

### *Slides in tablature*

Slides can be typeset in both `Staff` and `TabStaff` contexts:

```

slides = {
  c'8\3(\glissando d'8\3)
  c'8\3\glissando d'8\3
  \hideNotes
  \grace { g16\glissando }
  \unHideNotes
  c'4\3
  \afterGrace d'4\3\glissando {
    \stemDown \hideNotes

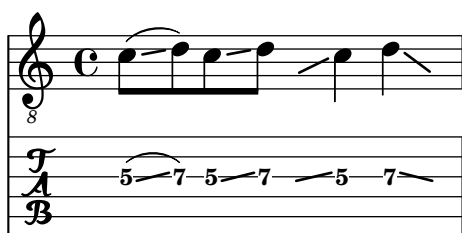
```

```

g16 }
\unHideNotes
}

\score {
  <<
    \new Staff { \clef "treble_8" \slides }
    \new TabStaff { \slides }
  >>
  \layout {
    \context {
      \Score
      \override Glissando.minimum-length = #4
      \override Glissando.springs-and-rods =
        #ly:spanner::set-spacing-rods
      \override Glissando.thickness = #2
      \omit StringNumber
      % or:
      %\override StringNumber.stencil = ##f
    }
  }
}

```



### *Chord glissando in tablature*

Slides for chords are indicated by default in both `Staff` and `TabStaff`. String numbers are necessary for `TabStaff` because automatic string calculations are different for chords and for single notes.

```

myMusic = \relative c' {
  <c e g>1 \glissando <f a c>
}

```

```

\score {
  <<
    \new Staff {
      \clef "treble_8"
      \myMusic
    }
    \new TabStaff \myMusic
  >>
}

```

```

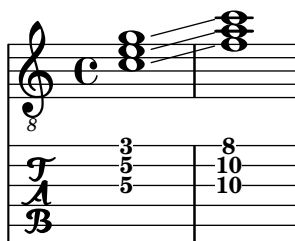
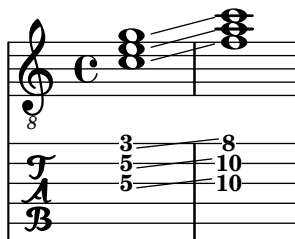
\score {
  <<
    \new Staff {

```

```

\clef "treble_8"
\myMusic
}
\new TabStaff \with { \override Glissando.style = #'none } {
  \myMusic
}
>>
}

```



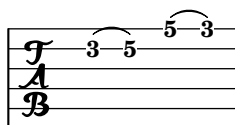
### *Hammer on and pull off*

Hammer-on and pull-off can be obtained using slurs.

```

\new TabStaff {
  \relative c' {
    d4( e\2)
    a( g)
  }
}

```



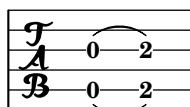
### *Hammer on and pull off using voices*

The arc of hammer-on and pull-off is upwards in voices one and three and downwards in voices two and four:

```

\new TabStaff {
  \relative c' {
    << { \voiceOne g2( a) }
    \\\ { \voiceTwo a,( b) }
    >> \oneVoice
  }
}

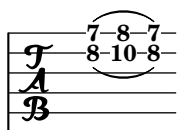
```



*Hammer on and pull off using chords*

When using hammer-on or pull-off with chorded notes, only a single arc is drawn. However “double arcs” are possible by setting the `doubleSlurs` property to `#t`.

```
\new TabStaff {
  \relative c' {
    % chord hammer-on and pull-off
    \set doubleSlurs = ##t
    <g' b>8( <a c> <g b>)
  }
}
```

**Vedi anche**

Notation Reference: [Chord repetition](#), pagina [Glissando](#), pagina 140, [Harmonics](#), pagina 337, [Stems](#), pagina [Written-out repeats](#), pagina [Written-out repeats](#).

Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

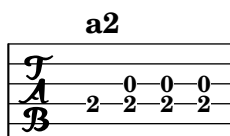
Internals Reference: Sezione “TabNoteHead” in *Guida al Funzionamento Interno*, Sezione “TabStaff” in *Guida al Funzionamento Interno*, Sezione “TabVoice” in *Guida al Funzionamento Interno*, Sezione “Beam” in *Guida al Funzionamento Interno*.

**Problemi noti e avvertimenti**

Chords are not handled in a special way, and hence the automatic string selector may easily select the same string for two notes in a chord.

In order to handle `\partcombine`, a `TabStaff` must use specially-created voices:

```
melodia = \partcombine { e4 g g g } { e4 e e e }
<<
  \new TabStaff <<
    \new TabVoice = "one" s1
    \new TabVoice = "two" s1
    \new TabVoice = "shared" s1
    \new TabVoice = "solo" s1
    { \melodia }
  >>
>>
```



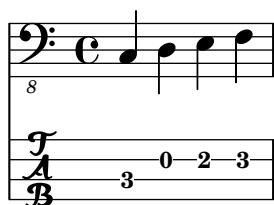
Guitar special effects are limited to harmonics and slides.

## Custom tablatures

LilyPond tablature automatically calculates the fret for a note based on the string to which the note is assigned. In order to do this, the tuning of the strings must be specified. The tuning of the strings is given in the `stringTunings` property.

LilyPond comes with predefined string tunings for banjo, mandolin, guitar, bass guitar, ukulele, violin, viola, cello, and double bass. LilyPond automatically sets the correct transposition for predefined tunings. The following example is for bass guitar, which sounds an octave lower than written.

```
<<
\new Voice \with {
  \omit StringNumber
} {
  \clef "bass_8"
  \relative {
    c,4 d e f
  }
}
\new TabStaff \with {
  stringTunings = #bass-tuning
} {
  \relative {
    c,4 d e f
  }
}
>>
```



The default string tuning is `guitar-tuning`, which is the standard EAD-GBE tuning. Some other predefined tunings are `guitar-open-g-tuning`, `mandolin-tuning` and `banjo-open-g-tuning`. The predefined string tunings are found in `ly/string-tunings-init.ly`.

Any desired string tuning can be created. The `\stringTuning` function can be used to define a string tuning which can be used to set `stringTunings` for the current context.

Its argument is a chord construct defining the pitches of each string in the tuning. The chord construct must be in absolute octave mode, see [\[Absolute octave entry\]](#), pagina [\(undefined\)](#). The string with the highest number (generally the lowest string) must come first in the chord. For example, we can define a string tuning for a four-string instrument with pitches of `a''`, `d''`, `g'`, and `c'`:

```
mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
}
```

```
<<
```

```

\new Staff {
  \clef treble
  \mynotes
}
\new TabStaff {
  \set Staff.stringTunings = \stringTuning <c' g' d'' a''>
  \mynotes
}
>>

```



The `stringTunings` property is also used by `FretBoards` to calculate automatic fret diagrams.

String tunings are used as part of the hash key for predefined fret diagrams (see [Predefined fret diagrams], pagina 369).

The previous example could also be written as follows:

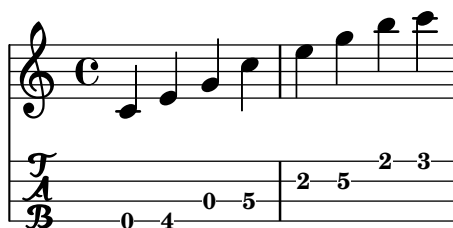
```

custom-tuning = \stringTuning <c' g' d'' a''>

mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
}

<<
\new Staff {
  \clef treble
  \mynotes
}
\new TabStaff {
  \set TabStaff.stringTunings = #custom-tuning
  \mynotes
}
>>

```



Internally, a string tuning is a Scheme list of string pitches, one for each string, ordered by string number from 1 to N, where string 1 is at the top of the tablature staff and string N is at the bottom. This ordinarily results in ordering from highest pitch to lowest pitch, but some instruments (e.g. ukulele) do not have strings ordered by pitch.

A string pitch in a string tuning list is a LilyPond pitch object. Pitch objects are created with the Scheme function `ly:make-pitch` (see [\[Scheme functions\]](#), pagina [\[undefined\]](#)).

`\stringTuning` creates such an object from chord input.

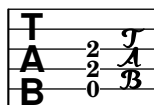
LilyPond automatically calculates the number of lines in the `TabStaff` and the number of strings in an automatically calculated `FretBoard` as the number of elements in `stringTunings`.

To let all `TabStaff` contexts use the same custom tuning by default, you can use

```
\layout {
  \context {
    \TabStaff
    stringTunings = \stringTuning <c' g' d'' a''>
  }
}
```

A modern tab clef can also be used.

```
\new TabStaff {
  \clef moderntab
  <a, e a>1
  \break
  \clef tab
  <a, e a>1
}
```



2



The modern tab clef supports tablatures from 4 to 7 strings.

`TabStaff` may support micro-tones like quarter-tones, which can be played using bendings. `supportNonIntegerFret = ##t` needs to be set in `Score`-context. However, micro-tones are not supported in `FretBoards`.

```
\layout {
  \context {
    \Score
    supportNonIntegerFret = ##t
  }
}
```

```
custom-tuning = \stringTuning <e, a, d ges beh eeh'>
```

```
mus = \relative {
  eeses'4
  eeseh
  ees
  eeh
  e
  eih
}
```

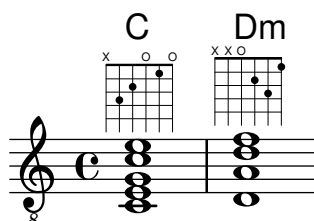




```

<d a d' f'>1^\markup {
  \fret-diagram #"6-x;5-x;4-o;3-2;2-3;1-1;"
}
}
>>

```

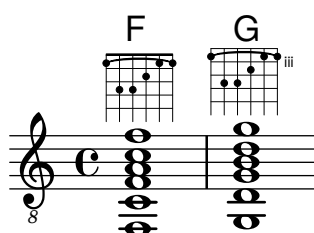


Barre indications can be added to the diagram from the fret-diagram markup string.

```

<<
\new ChordNames {
  \chordmode {
    f1 g
  }
}
\new Staff {
  \clef "treble_8"
  <f, c f a c' f'>1^\markup {
    \fret-diagram #"c:6-1-1;6-1;5-3;4-3;3-2;2-1;1-1;"
  }
  <g, d g b d' g'>1^\markup {
    \fret-diagram #"c:6-1-3;6-3;5-5;4-5;3-4;2-3;1-3;"
  }
}
>>

```



The size of the fret diagram, and the number of frets in the diagram can be changed in the fret-diagram markup string.

```

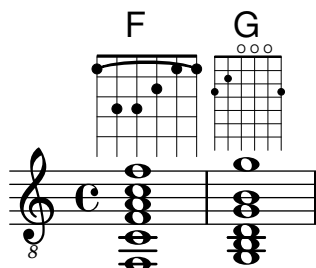
<<
\new ChordNames {
  \chordmode {
    f1 g
  }
}
\new Staff {
  \clef "treble_8"
  <f, c f a c' f'>1^\markup {
    \fret-diagram #"s:1.5;c:6-1-1;6-1;5-3;4-3;3-2;2-1;1-1;"
  }
}

```

```

<g, b, d g b g'>1^\markup {
  \fret-diagram #"h:6;6-3;5-2;4-o;3-o;2-o;1-3;"
}
}
>>

```

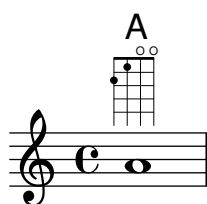


The number of strings in a fret diagram can be changed to accommodate different instruments such as banjos and ukuleles with the fret-diagram markup string.

```

<<
\new ChordNames {
  \chordmode {
    a1
  }
}
\new Staff {
  % An 'A' chord for ukulele
  a'1^\markup {
    \fret-diagram #"w:4;4-2-2;3-1-1;2-o;1-o;"
  }
}
>>

```



Fingering indications can be added, and the location of fingering labels can be controlled by the fret-diagram markup string.

```

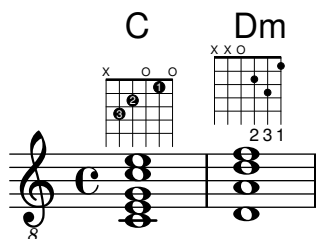
<<
\new ChordNames {
  \chordmode {
    c1 d:m
  }
}
\new Staff {
  \clef "treble_8"
  <c e g c' e'>1^\markup {
    \fret-diagram #"f:1;6-x;5-3-3;4-2-2;3-o;2-1-1;1-o;"
  }
  <d a d' f'>1^\markup {

```

```

\fret-diagram #"f:2;6-x;5-x;4-o;3-2-2;2-3-3;1-1-1;"
}
}
>>

```

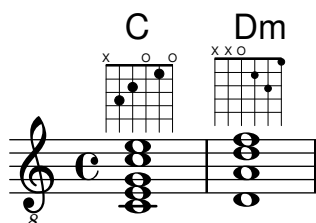


Dot radius and dot position can be controlled with the fret-diagram markup string.

```

<<
\new ChordNames {
  \chordmode {
    c1 d:m
  }
}
\new Staff {
  \clef "treble_8"
  <c e g c' e'>1^\markup {
    \fret-diagram #"d:0.35;6-x;5-3;4-2;3-o;2-1;1-o;"
  }
  <d a d' f'>1^\markup {
    \fret-diagram #"p:0.2;6-x;5-x;4-o;3-2;2-3;1-1;"
  }
}
>>

```



The fret-diagram-terse markup string omits string numbers; the string number is implied by the presence of semicolons. There is one semicolon for each string in the diagram. The first semicolon corresponds to the highest string number and the last semicolon corresponds to the first string. Mute strings, open strings, and fret numbers can be indicated.

```

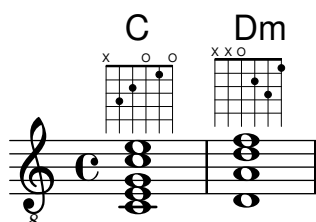
<<
\new ChordNames {
  \chordmode {
    c1 d:m
  }
}
\new Staff {
  \clef "treble_8"
  <c e g c' e'>1^\markup {
    \fret-diagram-terse #"x;3;2;o;1;o;"
  }
}
>>

```

```

    }
    <d a d' f'>1^\markup {
      \fret-diagram-terse #"x;x;o;2;3;1;"
    }
  }
}
>>

```

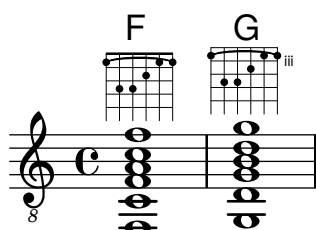


Barre indicators can be included in the fret-diagram-terse markup string.

```

<<
\new ChordNames {
  \chordmode {
    f1 g
  }
}
\new Staff {
  \clef "treble_8"
  <f, c f a c' f'>1^\markup {
    \fret-diagram-terse #"1-(;3;3;2;1;1-);"
  }
  <g, d g b d' g'>1^\markup {
    \fret-diagram-terse #"3-(;5;5;4;3;3-);"
  }
}
>>

```



Fingering indications can be included in the fret-diagram-terse markup string.

```

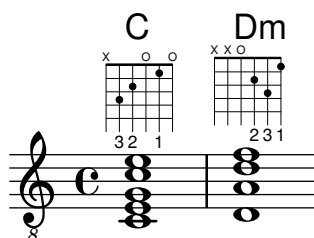
<<
\new ChordNames {
  \chordmode {
    c1 d:m
  }
}
\new Staff {
  \override Voice.TextScript.fret-diagram-details.finger-code = #'below-string
  \clef "treble_8"
  <c e g c' e'>1^\markup {
    \fret-diagram-terse #"x;3-3;2-2;o;1-1;o;"
  }
}
>>

```

```

    }
    <d a d' f'>1^\markup {
      \fret-diagram-terse #"x;x;o;2-2;3-3;1-1;"
    }
  }
}
>>

```



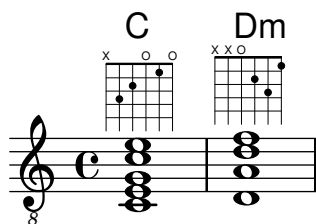
Other fret diagram properties must be adjusted using `\override` when using the `fret-diagram-terse` markup.

The `fret-diagram-verbose` markup string is in the format of a Scheme list. Each element of the list indicates an item to be placed on the fret diagram.

```

<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-verbose #'(
        (mute 6)
        (place-fret 5 3)
        (place-fret 4 2)
        (open 3)
        (place-fret 2 1)
        (open 1)
      )
    }
    <d a d' f'>1^\markup {
      \fret-diagram-verbose #'(
        (mute 6)
        (mute 5)
        (open 4)
        (place-fret 3 2)
        (place-fret 2 3)
        (place-fret 1 1)
      )
    }
  }
}
>>

```



Fingering indications and barres can be included in a fret-diagram-verbose markup string. Unique to the fret-diagram-verbose interface is a capo indication that can be placed on the fret diagram. The capo indication is a thick bar that covers all strings. The fret with the capo will be the lowest fret in the fret diagram.

Fingering indication dots can be colored as well as parenthesized; the parenthesis's color can also be altered independently.

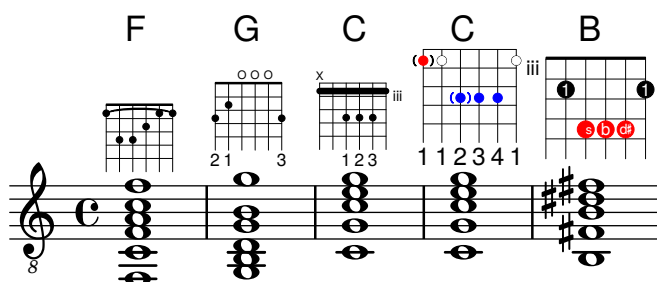
Markups can be placed into the dots as well.

```
<<
  \new ChordNames {
    \chordmode {
      f1 g c c b
    }
  }
  \new Staff {
    \clef "treble_8"
    \override Voice.TextScript.fret-diagram-details.finger-code = #'below-string
    <f, c f a c' f'>1^\markup {
      \fret-diagram-verbose #'(
        (place-fret 6 1)
        (place-fret 5 3)
        (place-fret 4 3)
        (place-fret 3 2)
        (place-fret 2 1)
        (place-fret 1 1)
        (barre 6 1 1)
      )
    }
    <g, b, d g b g'>1^\markup {
      \fret-diagram-verbose #'(
        (place-fret 6 3 2)
        (place-fret 5 2 1)
        (open 4)
        (open 3)
        (open 2)
        (place-fret 1 3 3)
      )
    }
    <c g c' e' g'>1^\markup {
      \fret-diagram-verbose #'(
        (capo 3)
        (mute 6)
        (place-fret 4 5 1)
        (place-fret 3 5 2)
        (place-fret 2 5 3)
      )
    }
  }
  \override Voice.TextScript.size = 1.4
```

```

<c g c' e' g'>1^\markup {
  \fret-diagram-verbose #'(
    (place-fret 6 3 1 red parenthesized default-paren-color)
    (place-fret 5 3 1 inverted)
    (place-fret 4 5 2 blue parenthesized)
    (place-fret 3 5 3 blue)
    (place-fret 2 5 4 blue)
    (place-fret 1 3 1 inverted)
  )
}
\override Voice.TextScript.size = 1.5
<b, fis b dis' fis'>1^\markup {
  \override #'(fret-diagram-details . ((finger-code . in-dot)))
  \fret-diagram-verbose #'(
    (place-fret 5 2 1)
    (place-fret 4 4 "fis" red)
    (place-fret 3 4 "b" red)
    (place-fret
      2 4
      ,#{ \markup
        \concat {
          \vcenter "d"
          \fontsize #-5
          \musicglyph #"accidentals.sharp"} #}
      red)
    (place-fret 1 2 1)
  )
}
}
>>

```



All other fret diagram properties must be adjusted using `\override` when using the `fret-diagram-verbose` markup.

The graphical layout of a fret diagram can be customized according to user preference through the properties of the `fret-diagram-interface`. Details are found at Sezione “fret-diagram-interface” in *Guida al Funzionamento Interno*. For a fret diagram markup, the interface properties belong to `Voice.TextScript`.

## Frammenti di codice selezionati

### *Changing fret orientations*

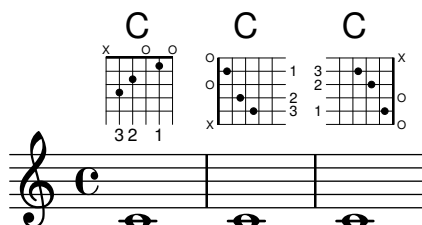
Fret diagrams can be oriented in three ways. By default the top string or fret in the different orientations will be aligned.

```
\include "predefined-guitar-fretboards.ly"
```

```

<<
\chords {
  c1
  c1
  c1
}
\new FretBoards {
  \chordmode {
    c1
    \override FretBoard.fret-diagram-details.orientation =
      #'landscape
    c1
    \override FretBoard.fret-diagram-details.orientation =
      #'opposing-landscape
    c1
  }
}
\new Voice {
  c'1
  c'1
  c'
}
>>

```



### *Customizing markup fret diagrams*

Fret diagram properties can be set through 'fret-diagram-details. For markup fret diagrams, overrides can be applied to the `Voice.TextScript` object or directly to the markup.

```

<<
\chords { c1 | c | c | d }

\new Voice = "mel" {
  \textLengthOn
  % Set global properties of fret diagram
  \override TextScript.size = #'1.2
  \override TextScript.fret-diagram-details.finger-code = #'in-dot
  \override TextScript.fret-diagram-details.dot-color = #'white

  %% C major for guitar, no barre, using defaults
  % terse style
  c'1^\markup { \fret-diagram-terse #"x;3-3;2-2;o;1-1;o;" }

  %% C major for guitar, barred on third fret
  % verbose style

```



```

% size 1.0
% roman fret label, finger labels below string, straight barre
c'1^\markup {
  % standard size
  \override #'(size . 1.0) {
    \override #'(fret-diagram-details . (
      (number-type . roman-lower)
      (finger-code . in-dot)
      (barre-type . straight))) {
      \fret-diagram-verbose #'((mute 6)
        (place-fret 5 3 1)
        (place-fret 4 5 2)
        (place-fret 3 5 3)
        (place-fret 2 5 4)
        (place-fret 1 3 1)
        (barre 5 1 3))
    }
  }
}

%% C major for guitar, barred on third fret
% verbose style
% landscape orientation, arabic numbers, M for mute string
% no barre, fret label down or left, small mute label font
c'1^\markup {
  \override #'(fret-diagram-details . (
    (finger-code . below-string)
    (number-type . arabic)
    (label-dir . -1)
    (mute-string . "M")
    (orientation . landscape)
    (barre-type . none)
    (xo-font-magnification . 0.4)
    (xo-padding . 0.3))) {
    \fret-diagram-verbose #'((mute 6)
      (place-fret 5 3 1)
      (place-fret 4 5 2)
      (place-fret 3 5 3)
      (place-fret 2 5 4)
      (place-fret 1 3 1)
      (barre 5 1 3))
    }
  }
}

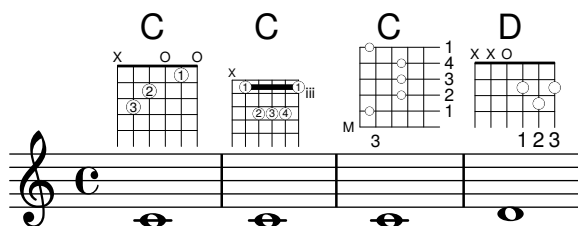
%% simple D chord
% terse style
% larger dots, centered dots, fewer frets
% label below string
d'1^\markup {
  \override #'(fret-diagram-details . (
    (finger-code . below-string)
    (dot-radius . 0.35)

```

```

(dot-position . 0.5)
(fret-count . 3))) {
\ fret-diagram-terse #"x;x;o;2-1;3-2;2-3;"
}
}
}
>>

```



## Vedi anche

Notation Reference: Sezione A.11.5 [Instrument Specific Markup], pagina 722.

Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

Internals Reference: Sezione “fret-diagram-interface” in *Guida al Funzionamento Interno*.

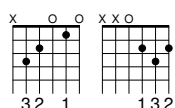
## Predefined fret diagrams

Fret diagrams can be displayed using the `FretBoards` context. By default, the `FretBoards` context will display fret diagrams that are stored in a lookup table:

```

\include "predefined-guitar-fretboards.ly"
\new FretBoards {
  \chordmode {
    c1 d
  }
}

```



The default predefined fret diagrams are contained in the file `predefined-guitar-fretboards.ly`. Fret diagrams are stored based on the pitches of a chord and the value of `stringTunings` that is currently in use. `predefined-guitar-fretboards.ly` contains predefined fret diagrams only for `guitar-tuning`. Predefined fret diagrams can be added for other instruments or other tunings by following the examples found in `predefined-guitar-fretboards.ly`.

Fret diagrams for the ukulele are contained in the file `predefined-ukulele-fretboards.ly`.

```

\include "predefined-ukulele-fretboards.ly"

```

```

myChords = \chordmode { a1 a:m a:aug }

```

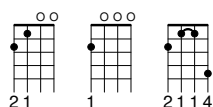
```

\new ChordNames {
  \myChords
}

```

```
\new FretBoards {
  \set Staff.stringTunings = #ukulele-tuning
  \myChords
}
```

A   Am   A+



Fret diagrams for the mandolin are contained in the file `predefined-mandolin-fretboards.ly`.

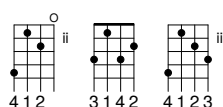
```
\include "predefined-mandolin-fretboards.ly"
```

```
myChords = \chordmode { c1 c:m7.5- c:aug }
```

```
\new ChordNames {
  \myChords
}
```

```
\new FretBoards {
  \set Staff.stringTunings = #mandolin-tuning
  \myChords
}
```

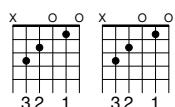
C   C<sup>∅</sup>   C+



Chord pitches can be entered either as simultaneous music or using chord mode (see [Chord mode overview], pagina 411).

```
\include "predefined-guitar-fretboards.ly"
```

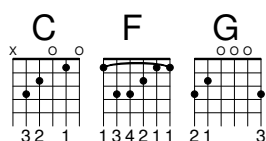
```
\new FretBoards {
  \chordmode { c1 }
  <c' e' g'>1
}
```



It is common that both chord names and fret diagrams are displayed together. This is achieved by putting a `ChordNames` context in parallel with a `FretBoards` context and giving both contexts the same music.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode{
  c1 f g
}
```

```
<<
\new ChordNames {
  \mychords
}
\new FretBoards {
  \mychords
}
>>
```

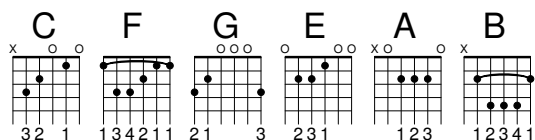


Predefined fret diagrams are transposable, as long as a diagram for the transposed chord is stored in the fret diagram table.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode{
  c1 f g
}
```

```
mychordlist = {
  \mychords
  \transpose c e { \mychords }
}
```

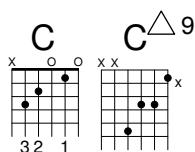
```
<<
\new ChordNames {
  \mychordlist
}
\new FretBoards {
  \mychordlist
}
>>
```



The predefined fret diagram table for guitar contains eight chords (major, minor, augmented, diminished, dominant seventh, major seventh, minor seventh, dominant ninth) for each of 17 keys. The predefined fret diagram table for ukulele contains these chords plus an additional three chords (major sixth, suspended second, and suspended fourth). A complete list of the predefined fret diagrams is shown in [\[Predefined fretboard diagrams\]](#), pagina [\[undefined\]](#). If there is no entry in the table for a chord, the FretBoards engraver will calculate a fret-diagram using the automatic fret diagram functionality described in [\[Automatic fret diagrams\]](#), pagina 379.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode{
  c1 c:maj9
}
```

```
<<
\new ChordNames {
  \mychords
}
\new FretBoards {
  \mychords
}
>>
```



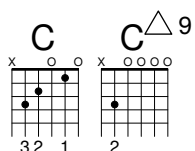
Fret diagrams can be added to the fret diagram table. To add a diagram, you must specify the hash table for the diagram, the chord for the diagram, the tuning to be used, and a definition for the diagram. Normally, the hash table will be *default-fret-table*. The diagram definition can be either a fret-diagram-terse definition string or a fret-diagram-verbose marking list.

```
\include "predefined-guitar-fretboards.ly"

\storePredefinedDiagram #default-fret-table
                        \chordmode { c:maj9 }
                        #guitar-tuning
                        #"x;3-2;o;o;o;o;"

mychords = \chordmode {
  c1 c:maj9
}
```

```
<<
\new ChordNames {
  \mychords
}
\new FretBoards {
  \mychords
}
>>
```



Different fret diagrams for the same chord name can be stored using different octaves of pitches. The different octave should be at least two octaves above or below the default octave, because the octaves above and below the default octave are used for transposing fretboards.

```
\include "predefined-guitar-fretboards.ly"

\storePredefinedDiagram #default-fret-table
                        \chordmode { c'' }
```

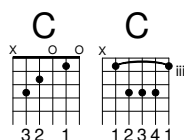
```

#guitar-tuning
#(offset-fret 2 (chord-shape 'bes guitar-tuning))

mychords = \chordmode {
  c1 c''
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>

```



In addition to fret diagrams, LilyPond stores an internal list of chord shapes. The chord shapes are fret diagrams that can be shifted along the neck to different positions to provide different chords. Chord shapes can be added to the internal list and then used to define predefined fret diagrams. Because they can be moved to various positions on the neck, chord shapes will normally not contain any open strings. Like fret diagrams, chord shapes can be entered as either fret-diagram-terse strings or fret-diagram-verbose marking lists.

```

\include "predefined-guitar-fretboards.ly"

% Add a new chord shape

\addChordShape #'powerf #guitar-tuning #"1-1;3-3;3-4;x;x;x;"

% add some new chords based on the power chord shape

\storePredefinedDiagram #default-fret-table
  \chordmode { f'' }
  #guitar-tuning
  #(chord-shape 'powerf guitar-tuning)
\storePredefinedDiagram #default-fret-table
  \chordmode { g'' }
  #guitar-tuning
  #(offset-fret 2 (chord-shape 'powerf guitar-tuning))

mychords = \chordmode{
  f1 f'' g g''
}

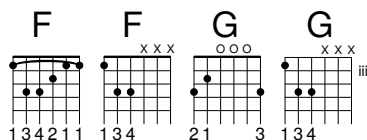
<<
  \new ChordNames {
    \mychords
  }

```

```

\new FretBoards {
  \mychords
}
>>

```



The graphical layout of a fret diagram can be customized according to user preference through the properties of the `fret-diagram-interface`. Details are found at Sezione “fret-diagram-interface” in *Guida al Funzionamento Interno*. For a predefined fret diagram, the interface properties belong to `FretBoards.FretBoard`.

## Frammenti di codice selezionati

### *Customizing fretboard fret diagrams*

Fret diagram properties can be set through '`fret-diagram-details`'. For FretBoard fret diagrams, overrides are applied to the `FretBoards.FretBoard` object. Like Voice, `FretBoards` is a bottom level context, therefore can be omitted in property overrides.

```

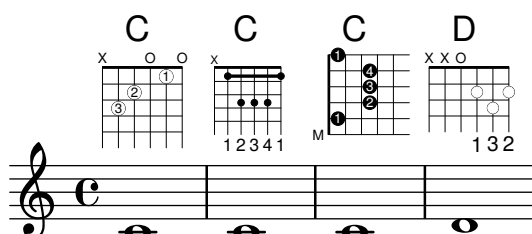
\include "predefined-guitar-fretboards.ly"
\storePredefinedDiagram #default-fret-table \chordmode { c' }
    #guitar-tuning
    #"x;1-1-(;3-2;3-3;3-4;1-1-);"
<<
\new ChordNames {
  \chordmode { c1 | c | c | d }
}
\new FretBoards {
  % Set global properties of fret diagram
  \override FretBoards.FretBoard.size = #'1.2
  \override FretBoard.fret-diagram-details.finger-code = #'in-dot
  \override FretBoard.fret-diagram-details.dot-color = #'white
  \chordmode {
    c
    \once \override FretBoard.size = #'1.0
    \once \override FretBoard.fret-diagram-details.barre-type = #'straight
    \once \override FretBoard.fret-diagram-details.dot-color = #'black
    \once \override FretBoard.fret-diagram-details.finger-code = #'below-string
    c'
    \once \override FretBoard.fret-diagram-details.barre-type = #'none
    \once \override FretBoard.fret-diagram-details.number-type = #'arabic
    \once \override FretBoard.fret-diagram-details.orientation = #'landscape
    \once \override FretBoard.fret-diagram-details.mute-string = #'M"
    \once \override FretBoard.fret-diagram-details.label-dir = #LEFT
    \once \override FretBoard.fret-diagram-details.dot-color = #'black
    c'
    \once \override FretBoard.fret-diagram-details.finger-code = #'below-string
    \once \override FretBoard.fret-diagram-details.dot-radius = #0.35
    \once \override FretBoard.fret-diagram-details.dot-position = #0.5
    \once \override FretBoard.fret-diagram-details.fret-count = #3
    d
  }
}

```

```

    }
  }
  \new Voice {
    c'1 | c' | c' | d'
  }
>>

```



### *Defining predefined fretboards for other instruments*

Predefined fret diagrams can be added for new instruments in addition to the standards used for guitar. This file shows how this is done by defining a new string-tuning and a few predefined fretboards for the Venezuelan cuatro.

This file also shows how fingerings can be included in the chords used as reference points for the chord lookup, and displayed in the fret diagram and the `TabStaff`, but not the music.

These fretboards are not transposable because they contain string information. This is planned to be corrected in the future.

```

% add FretBoards for the Cuatro
% Note: This section could be put into a separate file
% predefined-cuatro-fretboards.ly
% and \included into each of your compositions

```

```

cuatroTuning = #`(,(ly:make-pitch 0 6 0)
                  ,(ly:make-pitch 1 3 SHARP)
                  ,(ly:make-pitch 1 1 0)
                  ,(ly:make-pitch 0 5 0))

```

```

dSix = { <a\4 b\1 d\3 fis\2> }
dMajor = { <a\4 d\1 d\3 fis \2> }
aMajSeven = { <a\4 cis\1 e\3 g\2> }
dMajSeven = { <a\4 c\1 d\3 fis\2> }
gMajor = { <b\4 b\1 d\3 g\2> }

```

```

\storePredefinedDiagram #default-fret-table \dSix
                        #cuatroTuning
                        #"o;o;o;o;"
\storePredefinedDiagram #default-fret-table \dMajor
                        #cuatroTuning
                        #"o;o;o;3-3;"
\storePredefinedDiagram #default-fret-table \aMajSeven
                        #cuatroTuning
                        #"o;2-2;1-1;2-3;"
\storePredefinedDiagram #default-fret-table \dMajSeven
                        #cuatroTuning
                        #"o;o;o;1-1;"
\storePredefinedDiagram #default-fret-table \gMajor

```



```

#cuatroTuning
#"2-2;o;1-1;o;"

% end of potential include file /predefined-cuatro-fretboards.ly

#(set-global-staff-size 16)

primerosNames = \chordmode {
  d:6 d a:maj7 d:maj7
  g
}
primeros = {
  \dSix \dMajor \aMajSeven \dMajSeven
  \gMajor
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \primerosNames
    }

    \new Staff {
      \new Voice \with {
        \remove "New_fingering_engraver"
      }
      \relative c'' {
        \primeros
      }
    }

    \new FretBoards {
      \set Staff.stringTunings = #cuatroTuning
%      \override FretBoard
%      #'(fret-diagram-details string-count) = #'4
      \override FretBoard.fret-diagram-details.finger-code = #'in-dot
      \primeros
    }

    \new TabStaff \relative c'' {
      \set TabStaff.stringTunings = #cuatroTuning
      \primeros
    }

  >>

  \layout {
    \context {
      \Score
      \override SpacingSpanner.base-shortest-duration = #(ly:make-moment 1 16)
    }
  }
}

```

```

    }
  }
  \midi { }
}

```

	D <sup>6</sup>	D	A <sup>△</sup>	D <sup>△</sup>	G
T	0	3	2	1	0
M	0	0	1	0	1
B	0	0	2	0	0
B	0	0	0	0	2

### *ChordChanges for FretBoards*

FretBoards can be set to display only when the chord changes or at the beginning of a new line.

```
\include "predefined-guitar-fretboards.ly"
```

```

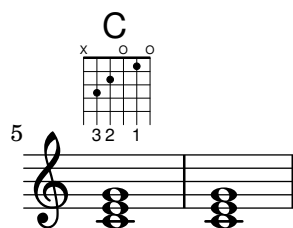
myChords = \chordmode {
  c1 c1 \break
  \set chordChanges = ##t
  c1 c1 \break
  c1 c1
}

```

```

<<
  \new ChordNames { \myChords }
  \new FretBoards { \myChords }
  \new Staff { \myChords }
>>

```



### *Fretboards alternate tables*

Alternate fretboard tables can be created. These would be used in order to have alternate fretboards for a given chord.

In order to use an alternate fretboard table, the table must first be created. Fretboards are then added to the table.

The created fretboard table can be blank, or it can be copied from an existing table.

The table to be used in displaying predefined fretboards is selected by the property `\predefinedDiagramTable`.

```
\include "predefined-guitar-fretboards.ly"
```

```
% Make a blank new fretboard table
```

```
 #(define custom-fretboard-table-one (make-fretboard-table))
```

```
% Make a new fretboard table as a copy of default-fret-table
```

```
 #(define custom-fretboard-table-two (make-fretboard-table default-fret-table))
```

```
% Add a chord to custom-fretboard-table-one
```

```
\storePredefinedDiagram #custom-fretboard-table-one
      \chordmode{c}
      #guitar-tuning
      "3-(;3;5;5;5;3-);"
```

```
% Add a chord to custom-fretboard-table-two
```

```
\storePredefinedDiagram #custom-fretboard-table-two
      \chordmode{c}
      #guitar-tuning
      "x;3;5;5;5;o;"
```

```
<<
```

```
\chords {
  c1 | d1 |
  c1 | d1 |
  c1 | d1 |
}
```

```
\new FretBoards {
```

```
  \chordmode {
    \set predefinedDiagramTable = #default-fret-table
    c1 | d1 |
    \set predefinedDiagramTable = #custom-fretboard-table-one
    c1 | d1 |
    \set predefinedDiagramTable = #custom-fretboard-table-two
    c1 | d1 |
  }
```

```
}
```

```
\new Staff {
  \clef "treble_8"
```

```

<<
  \chordmode {
    c1 | d1 |
    c1 | d1 |
    c1 | d1 |
  }
  {
    s1\_markup "Default table" | s1 |
    s1\_markup \column {"New table" "from empty"} | s1 |
    s1\_markup \column {"New table" "from default"} | s1 |
  }
>>
}
>>

```

Diagram illustrating three different chord tables (Default table, New table from empty, New table from default) for C and D chords, showing fretboard diagrams and corresponding musical notation.

## Vedi anche

Notation Reference: [Custom tablatures], pagina 356, [Automatic fret diagrams], pagina 379, [Chord mode overview], pagina 411, [\[Predefined fretboard diagrams\]](#), pagina [\[undefined\]](#).

Installed Files: [ly/predefined-guitar-fretboards.ly](#),  
[ly/predefined-guitar-ninth-fretboards.ly](#),  
[ly/predefined-ukulele-fretboards.ly](#),  
[ly/predefined-mandolin-fretboards.ly](#).

Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

Internals Reference: Sezione “fret-diagram-interface” in *Guida al Funzionamento Interno*.

## Automatic fret diagrams

Fret diagrams can be automatically created from entered notes using the **FretBoards** context. If no predefined diagram is available for the entered notes in the active **stringTunings**, this context calculates strings and frets that can be used to play the notes.

```

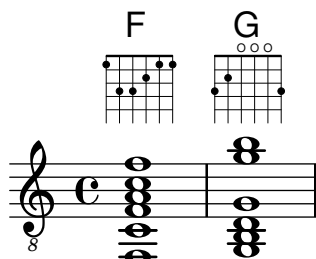
<<
  \new ChordNames {
    \chordmode {
      f1 g
    }
  }
  \new FretBoards {
    <f, c f a c' f'>1
    <g,\6 b, d g b g'>1
  }
  \new Staff {

```

```

\clef "treble_8"
<f, c f a c' f'>1
<g, b, d g b' g'>1
}
>>

```

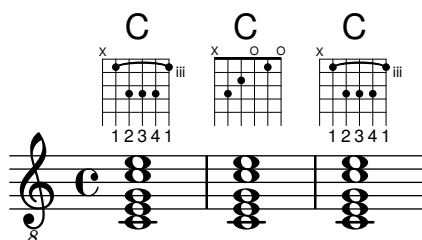


As no predefined diagrams are loaded by default, automatic calculation of fret diagrams is the default behavior. Once default diagrams are loaded, automatic calculation can be enabled and disabled with predefined commands:

```

\storePredefinedDiagram #default-fret-table
    <c e g c' e'>
    #guitar-tuning
    #"x;3-1-(;5-2;5-3;5-4;3-1-1-);"
<<
\new ChordNames {
  \chordmode {
    c1 c c
  }
}
\new FretBoards {
  <c e g c' e'>1
  \predefinedFretboardsOff
  <c e g c' e'>1
  \predefinedFretboardsOn
  <c e g c' e'>1
}
\new Staff {
  \clef "treble_8"
  <c e g c' e'>1
  <c e g c' e'>1
  <c e g c' e'>1
}
>>

```

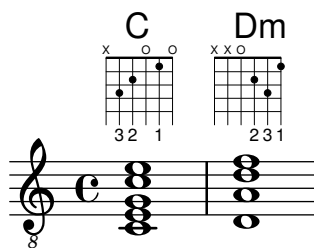


Sometimes the fretboard calculator will be unable to find an acceptable diagram. This can often be remedied by manually assigning a note to a string. In many cases, only one note need

be manually placed on a string; the rest of the notes will then be placed appropriately by the `FretBoards` context.

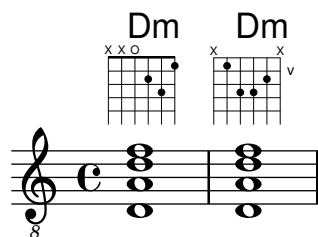
Fingerings can be added to `FretBoard` fret diagrams.

```
<<
\new ChordNames {
  \chordmode {
    c1 d:m
  }
}
\new FretBoards {
  <c-3 e-2 g c'-1 e'>1
  <d a-2 d'-3 f'-1>1
}
\new Staff {
  \clef "treble_8"
  <c e g c' e'>1
  <d a d' f'>1
}
>>
```



The minimum fret to be used in calculating strings and frets for the `FretBoard` context can be set with the `minimumFret` property.

```
<<
\new ChordNames {
  \chordmode {
    d1:m d:m
  }
}
\new FretBoards {
  <d a d' f'>1
  \set FretBoards.minimumFret = #5
  <d a d' f'>1
}
\new Staff {
  \clef "treble_8"
  <d a d' f'>1
  <d a d' f'>1
}
>>
```



The strings and frets for the `FretBoards` context depend on the `stringTunings` property, which has the same meaning as in the `TabStaff` context. See [Custom tablatures], pagina 356, for information on the `stringTunings` property.

The graphical layout of a fret diagram can be customized according to user preference through the properties of the `fret-diagram-interface`. Details are found at Sezione “fret-diagram-interface” in *Guida al Funzionamento Interno*. For a `FretBoards` fret diagram, the interface properties belong to `FretBoards.FretBoard`.

## Comandi predefiniti

`\predefinedFretboardsOff`, `\predefinedFretboardsOn`.

## Vedi anche

Notation Reference: [Custom tablatures], pagina 356.

Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

Internals Reference: Sezione “fret-diagram-interface” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Automatic fretboard calculations do not work properly for instruments with non-monotonic tunings.

## Right-hand fingerings

Right-hand fingerings *p-i-m-a* must be entered using `\rightHandFinger` followed by a number.

**Nota:** If the number is entered in Scheme notation, remember to append a space before following it with a closing `>` or similar.

```
\clef "treble_8"
c4\rightHandFinger #1
e\rightHandFinger #2
g\rightHandFinger #3
c'\rightHandFinger #4
<c\rightHandFinger #1 e\rightHandFinger #2
g\rightHandFinger #3 c'\rightHandFinger #4 >1
```



For convenience, you can abbreviate `\rightHandFinger` to something short, for example `RH`, `RH=#rightHandFinger`

## Frammenti di codice selezionati

*Placement of right-hand fingerings*

It is possible to exercise greater control over the placement of right-hand fingerings by setting a specific property, as demonstrated in the following example. Note: you must use a chord construct

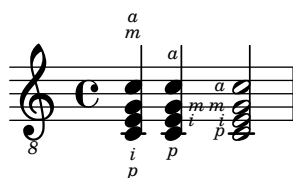
```
#(define RH rightHandFinger)

\relative c {
  \clef "treble_8"

  \set strokeFingerOrientations = #'(up down)
  <c-\RH #1 e-\RH #2 g-\RH #3 c-\RH #4 >4

  \set strokeFingerOrientations = #'(up right down)
  <c-\RH #1 e-\RH #2 g-\RH #3 c-\RH #4 >4

  \set strokeFingerOrientations = #'(left)
  <c-\RH #1 e-\RH #2 g-\RH #3 c-\RH #4 >2
}
```

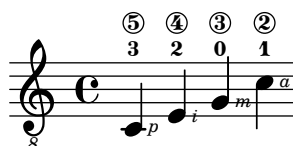


*Fingerings string indications and right-hand fingerings*

This example combines left-hand fingering, string indications, and right-hand fingering.

```
#(define RH rightHandFinger)

\relative c {
  \clef "treble_8"
  <c-3\5-\RH #1 >4
  <e-2\4-\RH #2 >4
  <g-0\3-\RH #3 >4
  <c-1\2-\RH #4 >4
}
```



## Vedi anche

Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

Internals Reference: Sezione “StrokeFinger” in *Guida al Funzionamento Interno*.

### 2.4.2 Guitar

Most of the notational issues associated with guitar music are covered sufficiently in the general fretted strings section, but there are a few more worth covering here. Occasionally users want to create songbook-type documents having only lyrics with chord indications above them. Since LilyPond is a music typesetter, it is not recommended for documents that have no music notation in them. A better alternative is a word processor, text editor, or, for experienced users, a typesetter like GuitarTeX.



## Indicating position and barring

This example demonstrates how to include guitar position and barring indications.

```
\relative {
  \clef "treble_8"
  b,16 d g b e
  \textSpannerDown
  \override TextSpanner.bound-details.left.text = #"XII "
  g16\startTextSpan
  b16 e g e b g\stopTextSpan
  e16 b g d
}
```



## Vedi anche

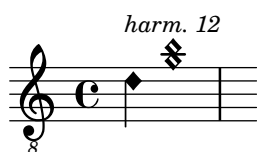
Notation Reference: [\[Text spanners\]](#), pagina [\[undefined\]](#).

Snippets: Sezione “Fretted strings” in *Frammenti di codice*, Sezione “Expressive marks” in *Frammenti di codice*.

## Indicating harmonics and dampened notes

Special note heads can be used to indicate dampened notes or harmonics. Harmonics are normally further explained with a text markup.

```
\relative {
  \clef "treble_8"
  \override Staff.NoteHead.style = #'harmonic-mixed
  d'^\markup { \italic { \fontsize #-2 { "harm. 12" }}} <g b>1
}
```



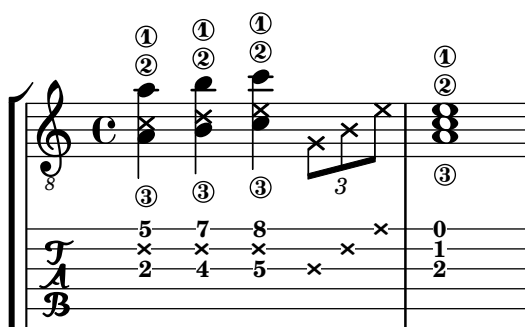
Dampened notes (also called *dead notes*) are supported within normal and tablature staves:

```
music = \relative {
  < a\3 \deadNote c\2 a'\1 >4
  < b\3 \deadNote d\2 b'\1 >
  < c\3 \deadNote e\2 c'\1 >
  \deadNotesOn
  \tuplet 3/2 { g8 b e }
  \deadNotesOff
  < a,\3 c\2 e\1 >1
}
\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
```

```

\music
}
\new TabStaff {
  \music
}
>>

```



Another playing technique (especially used on electric guitars) is called *palm mute*. The string is hereby partly muted by the palm of the striking hand (hence the name). Lilypond supports the notation of palm mute-style notes by changing the note head to a triangle shape.

```

\new Voice { % Warning: explicit Voice instantiation is
              %      required to have palmMuteOff work properly
              %      when palmMuteOn comes at the beginning of
              %      the piece.
\relative c, {
  \clef "G_8"
  \palmMuteOn
  e8~\markup { \musicglyph #"noteheads.u2do" = palm mute }
  < e b' e > e
  \palmMuteOff
  e e \palmMute e e e |
  e8 \palmMute { e e e } e e e e |
  < \palmMute e b' e >8 \palmMute { e e e } < \palmMute e b' e >2
}
}

```



## Vedi anche

Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

Notation Reference: [\[Special note heads\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Note head styles\]](#), pagina [\[undefined\]](#).

## Indicating power chords

Power chords and their symbols can be engraved in chord mode or as chord constructs:

```

ChordsAndSymbols = {
  \chordmode {

```

```

\powerChords
e,,1:5
a,,1:5.8
\set minimumFret = #8
c,1:5
f,1:5.8
}
\set minimumFret = #5
<a, e>1
<g d' g'>1
}
\score {
  <<
    \new ChordNames {
      \ChordsAndSymbols
    }
    \new Staff {
      \clef "treble_8"
      \ChordsAndSymbols
    }
    \new TabStaff {
      \ChordsAndSymbols
    }
  >>
}

```

The image displays a musical score for guitar. The top staff is a treble clef staff with a common time signature (C). It contains six measures, each with a power chord symbol above it: E<sup>5</sup>, A<sup>5</sup>, C<sup>5</sup>, F<sup>5</sup>, A<sup>5</sup>, and G<sup>5</sup>. The notes are represented by circles on the staff lines. Below the staff is a guitar tablature with six strings (E, A, D, G, B, E) and fret numbers. The fret numbers are: E string (8, 2, 0, 10, 7, 8), A string (2, 2, 0, 10, 7, 5), D string (2, 0, 8, 8, 7, 5), G string (0, 0, 8, 8, 7, 5), B string (2, 0, 8, 8, 7, 5), and E string (8, 2, 0, 10, 7, 8).

Power chord symbols are automatically switched off as soon as one of the other common chord modifier is used:

```

mixedChords = \chordmode {
  c,1
  \powerChords
  b,,1:5
  fis,,1:5.8
  g,,1:m
}
\score {
  <<
    \new ChordNames {
      \mixedChords
    }
    \new Staff {
      \clef "treble_8"

```

```

    \mixedChords
  }
  \new TabStaff {
    \mixedChords
  }
>>
}

```

	C	B <sup>5</sup>	F <sup>#5</sup>	Gm
Treble	0	4	4	0
Alto	2	2	4	1
Bass	3	2	2	3

## Vedi anche

Music Glossary: Sezione “power chord” in *Glossario Musicale*.

Notation Reference: [Extended and altered chords], pagina 413, [Printing chord names], pagina 416.

Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

## 2.4.3 Banjo

### Banjo tablatures

LilyPond has basic support for the five-string banjo. When making tablatures for five-string banjo, use the banjo tablature format function to get correct fret numbers for the fifth string:

```

music = {
  g8 d' g'\5 a b g e d' |
  g4 d''8\5 b' a'\2 g'\5 e'\2 d' |
  g4
}

<<
\new Staff \with { \omit StringNumber }
{ \clef "treble_8" \music }
\new TabStaff \with {
  tablatureFormat = #fret-number-tablature-format-banjo
  stringTunings = #banjo-open-g-tuning
}
{ \music }
>>

```

	Measure 1	Measure 2	Measure 3	Measure 4
Treble	0	9	10	5
Alto	0	2	0	0
Bass	0	2	12	0

A number of common tunings for the five-string banjo are predefined: `banjo-c-tuning` (gCGBD), `banjo-modal-tuning` (gDGCD), `banjo-open-d-tuning` (aDF#AD) and `banjo-open-dm-tuning` (aDFAD).

These may be converted to four-string tunings using the `four-string-banjo` function:

```
\set TabStaff.stringTunings = #(four-string-banjo banjo-c-tuning)
```

## Vedi anche

Installed Files: `ly/string-tunings-init.ly`.

Snippets: Sezione “Fretted strings” in *Frammenti di codice*.

## 2.5 Percussion

### 2.5.1 Common notation for percussion

Rhythmic music is primarily used for percussion and drum notation, but it can also be used to show the rhythms of melodies.

### References for percussion

- Some percussion may be notated on a rhythmic staff; this is discussed in [\[Showing melody rhythms\]](#), pagina [\[Instantiating new staves\]](#), pagina [\[MIDI instruments\]](#).
- MIDI output is discussed in a separate section; please see [\[MIDI instruments\]](#), pagina [\[Instantiating new staves\]](#).

## Vedi anche

Notation Reference: [\[Showing melody rhythms\]](#), pagina [\[Instantiating new staves\]](#), pagina [\[MIDI instruments\]](#).

Snippets: Sezione “Percussion” in *Frammenti di codice*.

## Basic percussion notation

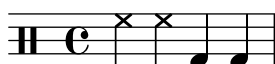
Percussion notes may be entered in `\drummode` mode, which is similar to the standard mode for entering notes. The simplest way to enter percussion notes is to use the `\drums` command, which creates the correct context and entry mode for percussion:

```
\drums {
  hihat4 hh bassdrum bd
}
```



This is shorthand for:

```
\new DrumStaff {
  \drummode {
    hihat4 hh bassdrum bd
  }
}
```



Each piece of percussion has a full name and an abbreviated name, and both can be used in input files. The full list of percussion note names may be found in [\[Percussion notes\]](#), pagina [\[undefined\]](#).

Note that the normal notation of pitches (such as `cis4`) in a `DrumStaff` context will cause an error message. Percussion clefs are added automatically to a `DrumStaff` context but they can also be set explicitly. Other clefs may be used as well.

```
\drums {
  \clef percussion
  bd4 4 4 4
  \clef treble
  hh4 4 4 4
}
```



There are a few issues concerning MIDI support for percussion instruments; for details please see [\[MIDI instruments\]](#), pagina [\[undefined\]](#).

## Vedi anche

Notation Reference: [\[undefined\]](#) [\[MIDI instruments\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Percussion notes\]](#), pagina [\[undefined\]](#).

Installed Files: `ly/drumpitch-init.ly`.

Snippets: Sezione “Percussion” in *Frammenti di codice*.

## Drum rolls

Drum rolls are indicated with three slashes across the stem. For quarter notes or longer the three slashes are shown explicitly, eighth notes are shown with two slashes (the beam being the third), and drum rolls shorter than eighths have one stem slash to supplement the beams. This is achieved with the tremolo notation, as described in [\[Tremolo repeats\]](#), pagina [\[undefined\]](#).

```
\drums {
  \time 2/4
  sn16 8 16 8 8:32 ~
  8 8 4:32 ~
  4 8 16 16
  4 r4
}
```



Sticking can be indicated by placing a markup for "R" or "L" above or below notes, as discussed in Sezione 5.4.2 [\[Direction and placement\]](#), pagina 611. The `staff-padding` property may be overridden to achieve a pleasing baseline.

```
\drums {
  \repeat unfold 2 {
    sn16~"L" 16~"R" 16~"L" 16~"L" 16~"R" 16~"L" 16~"R" 16~"R"
    \stemUp
    sn16_"L" 16_"R" 16_"L" 16_"L" 16_"R" 16_"L" 16_"R" 16_"R"
  }
```



## Vedi anche

Notation Reference: [\[Tremolo repeats\]](#), pagina [\[Tremolo repeats\]](#).

Snippets: Sezione “Percussion” in *Frammenti di codice*.

## Pitched percussion

Certain pitched percussion instruments (e.g. xylophone, vibraphone, and timpani) are written using normal staves. This is covered in other sections of the manual.

## Vedi anche

Notation Reference: [\[MIDI instruments\]](#), pagina [\[MIDI instruments\]](#).

Snippets: Sezione “Percussion” in *Frammenti di codice*.

## Percussion staves

A percussion part for more than one instrument typically uses a multiline staff where each position in the staff refers to one piece of percussion. To typeset the music, the notes must be interpreted in `DrumStaff` and `DrumVoice` context.

```
up = \drummode {
  crashcymbal4 hihat8 halfopenhihat hh hh hh openhihat
}
down = \drummode {
  bassdrum4 snare8 bd r bd sn4
}
\new DrumStaff <<
  \new DrumVoice { \voiceOne \up }
  \new DrumVoice { \voiceTwo \down }
>>
```



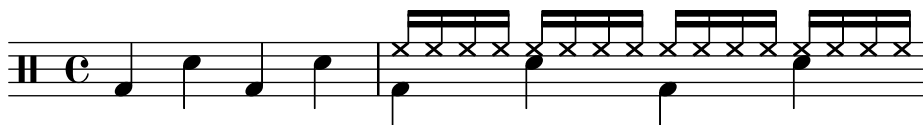
The above example shows verbose polyphonic notation. The short polyphonic notation, described in Sezione “I’m hearing Voices” in *Manuale di Apprendimento*, can also be used. For example,

```
\new DrumStaff <<
  \drummode {
    bd4 sn4 bd4 sn4
    << {
      \repeat unfold 16 hh16
    } \\ {
      bd4 sn4 bd4 sn4
    }
  }
>>
```

```

    } >>
  }
>>

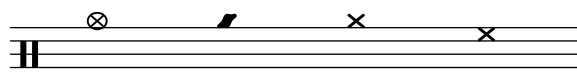
```



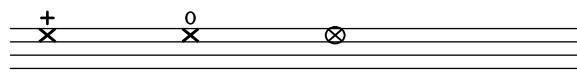
There are also other layout possibilities. To use these, set the property `drumStyleTable` in context `DrumVoice`. The following variables have been predefined:

#### drums-style

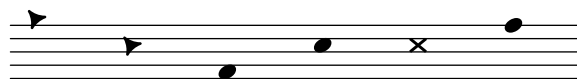
This is the default. It typesets a typical drum kit on a five-line staff:



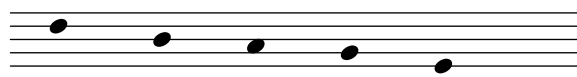
cymc cymc cymr hh



hhc hho hhho hhp



cb hc bd sn ss tomh

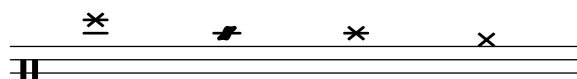


tommh tomml tomml tomfh tom

The drum scheme supports six different toms. When there are fewer toms, simply select the toms that produce the desired result. For example, to get toms on the three middle lines you use `tommh`, `tomml`, and `tomfh`.

#### agostini-drums-style

Invented by the French percussionist Dante Agostini in 1965, this notation is commonly employed in France but also elsewhere.

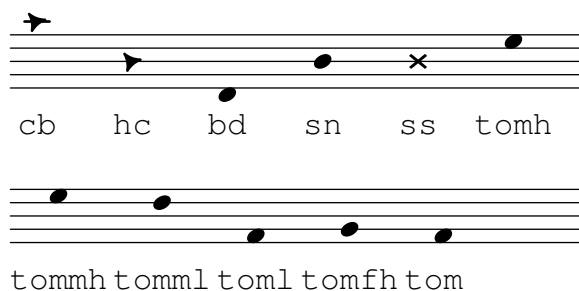


cymc cymc cymr hh

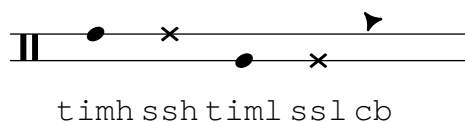


hhc hho hhho hhp

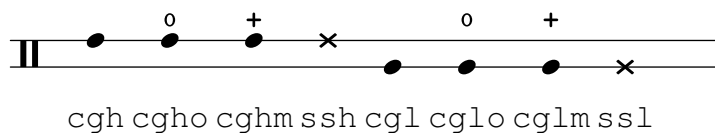


**timbales-style**

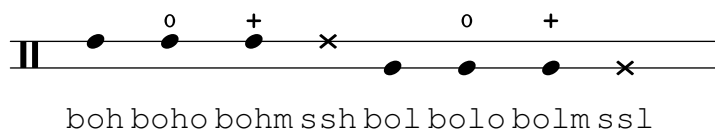
This typesets timbales on a two line staff:

**congas-style**

This typesets congas on a two line staff:

**bongos-style**

This typesets bongos on a two line staff:

**percussion-style**

To typeset all kinds of simple percussion on one line staves:

**Custom percussion staves**

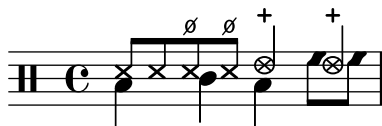
If you do not like any of the predefined lists you can define your own list at the top of your file.

```
#(define mydrums '(
  (bassdrum      default  #f      -1)
  (snare         default  #f      0)
  (hihat         cross    #f      1)
  (halfopenhihat cross    "halfopen" 1)
  (pedalhihat    xcircle  "stopped" 2)
  (lowtom        diamond  #f      3)))

up = \drummode { hh8 hh hhho hhho hhp4 hhp }
down = \drummode { bd4 sn bd toml8 toml }

\new DrumStaff <<
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \new DrumVoice { \voiceOne \up }
```

```
\new DrumVoice { \voiceTwo \down }
>>
```



## Frammenti di codice selezionati

Here are some examples:

Two Woodblocks, entered with wbh (high woodblock) and wbl (low woodblock)

```
% These lines define the position of the woodblocks in the stave;
% if you like, you can change it or you can use special note heads
% for the woodblocks.
#(define mydrums '((hiwoodblock default #t 3)
                    (lowwoodblock default #t -2)))

woodstaff = {
  % This defines a staff with only two lines.
  % It also defines the positions of the two lines.
  \override Staff.StaffSymbol.line-positions = #'(-2 3)

  % This is necessary; if not entered, the barline would be too short!
  \override Staff.BarLine.bar-extent = #'(-1.5 . 1.5)
}

\new DrumStaff {
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)

  % with this you load your new drum style table
  \woodstaff

  \drummode {
    \time 2/4
    wbl8 16 16 8-> 8 |
    wbl8 16 16-> ~ 16 16 r8 |
  }
}
```



Note that in this special case the length of the barline must be altered with `\override Staff.BarLine.bar-extent #'(from . to)`. Otherwise it would be too short. And you have also to define the positions of the two stafflines. For more information about these delicate things have a look at `[Staff symbol]`, pagina `[undefined]`.

A tambourine, entered with 'tamb':

```
#(define mydrums '((tambourine default #t 0)))

tambustaff = {
```

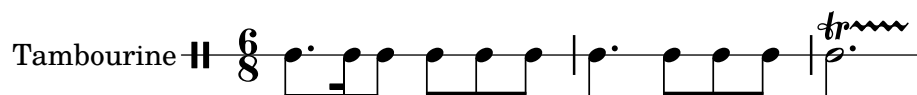
```

\override Staff.StaffSymbol.line-positions = #'( 0 )
\override Staff.BarLine.bar-extent = #'(-1.5 . 1.5)
\set DrumStaff.instrumentName = #"Tambourine"
}

\new DrumStaff {
  \tambustaff
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)

  \drummode {
    \time 6/8
    tamb8. 16 8 8 8 8 |
    tamb4. 8 8 8 |
    % the trick with the scaled duration and the shorter rest
    % is necessary for the correct ending of the trill-span!
    tamb2.*5/6 \startTrillSpan s8 \stopTrillSpan |
  }
}

```



Music for Tam-Tam (entered with 'tt'):

```

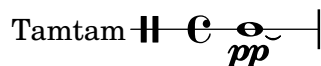
#(define mydrums '((tamtam default #t 0)))

tamtamstaff = {
  \override Staff.StaffSymbol.line-positions = #'( 0 )
  \override Staff.BarLine.bar-extent = #'(-1.5 . 1.5)
  \set DrumStaff.instrumentName = #"Tamtam"
}

\new DrumStaff {
  \tamtamstaff
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)

  \drummode {
    tt 1 \pp \laissezVibrer
  }
}

```



Two different bells, entered with 'cb' (cowbell) and 'rb' (ridebell)

```

#(define mydrums '((ridebell default #t 3)
                  (cowbell default #t -2)))

bellstaff = {
  \override DrumStaff.StaffSymbol.line-positions = #'(-2 3)
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \override Staff.BarLine.bar-extent = #'(-1.5 . 1.5)
}

```

```

\set DrumStaff.instrumentName = #"Different Bells"
}

\new DrumStaff {
  \bellstaff
  \drummode {
    \time 2/4
    rb8 8 cb8 16 rb16-> ~ |
    16 8 16 cb8 8 |
  }
}

```



Here a short example taken from Stravinsky's 'L'histoire du Soldat'.

```

#(define mydrums '((bassdrum default #t 4)
                    (snare default #t -4)
                    (tambourine default #t 0)))

global = {
  \time 3/8 s4.
  \time 2/4 s2*2
  \time 3/8 s4.
  \time 2/4 s2
}

drumsA = {
  \context DrumVoice <<
    { \global }
    { \drummode {
      \autoBeamOff
      \stemDown sn8 \stemUp tamb s8 |
      sn4 \stemDown sn4 |
      \stemUp tamb8 \stemDown sn8 \stemUp sn16 \stemDown sn \stemUp sn8 |
      \stemDown sn8 \stemUp tamb s8 |
      \stemUp sn4 s8 \stemUp tamb
    }
  }
  >>
}

drumsB = {
  \drummode {
    s4 bd8 s2*2 s4 bd8 s4 bd8 s8
  }
}

\layout {
  indent = #40
}

```

```

\score {
  \new StaffGroup <<
    \new DrumStaff {
      \set DrumStaff.instrumentName = \markup {
        \column {
          "Tambourine"
          "et"
          "caisse claire s. timbre"
        }
      }
      \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
      \drumsA
    }

    \new DrumStaff {
      \set DrumStaff.instrumentName = #"Grosse Caisse"
      \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
      \drumsB }
  >>
}

```

Tambourine  
et  
caisse claire s. timbre

Grosse Caisse



## Vedi anche

Snippets: Sezione “Percussion” in *Frammenti di codice*.

Internals Reference: Sezione “DrumStaff” in *Guida al Funzionamento Interno*, Sezione “DrumVoice” in *Guida al Funzionamento Interno*.

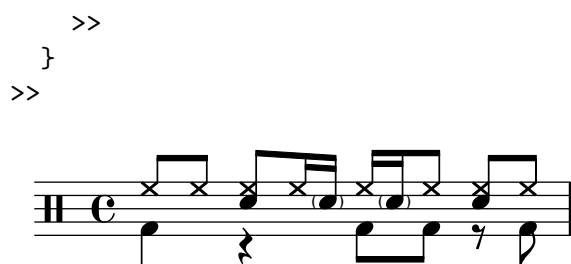
## Ghost notes

Ghost notes for drums and percussion may be created using the `\parenthesize` command detailed in [\[Parentheses\]](#), pagina [\[undefined\]](#).

```

\new DrumStaff
<<
  \context DrumVoice = "1" { s1 }
  \context DrumVoice = "2" { s1 }
  \drummode {
    <<
      {
        hh8[ 8] <hh sn> hh16
        \parenthesize sn hh
        \parenthesize sn hh8 <hh sn> hh
      } \
      {
        bd4 r4 bd8 8 r8 bd
      }
    }
  }

```



## Vedi anche

Snippets: Sezione “Percussion” in *Frammenti di codice*.

## 2.6 Wind instruments

**Moderato assai**

This section includes elements of music notation that arise when writing specifically for wind instruments.

### 2.6.1 Common notation for wind instruments

This section discusses notation common to most wind instruments.

## References for wind instruments

Many notation issues for wind instruments pertain to breathing and tonguing:

- Breathing can be specified by rests or `<undefined>` [Breath marks], pagina `<undefined>`.
- Legato playing is indicated by `<undefined>` [Slurs], pagina `<undefined>`.
- Different types of tonguings, ranging from legato to non-legato to staccato are usually shown by articulation marks, sometimes combined with slurs, see `<undefined>` [Articulations and ornamentations], pagina `<undefined>`, and `<undefined>` [List of articulations], pagina `<undefined>`.
- Flutter tonguing is usually indicated by placing a tremolo mark and a text markup on the note. See `<undefined>` [Tremolo repeats], pagina `<undefined>`.

Other aspects of musical notation that can apply to wind instruments:

- Many wind instruments are transposing instruments, see `<undefined>` [Instrument transpositions], pagina `<undefined>`.
- Slide glissandi are characteristic of the trombone, but other winds may perform keyed or valved glissandi. See [Glissando], pagina 140.
- Harmonic series glissandi, which are possible on all brass instruments but common for French Horns, are usually written out as `<undefined>` [Grace notes], pagina `<undefined>`.

- Pitch inflections at the end of a note are discussed in [\[Falls and doits\]](#), pagina [\[undefined\]](#).
- Key slaps or valve slaps are often shown by the `cross` style of [\[undefined\]](#) [\[Special note heads\]](#), pagina [\[undefined\]](#).
- Woodwinds can overblow low notes to sound harmonics. These are shown by the `flageolet` articulation. See [\[undefined\]](#) [\[List of articulations\]](#), pagina [\[undefined\]](#).
- The use of brass mutes is usually indicated by a text markup, but where there are many rapid changes it is better to use the `stopped` and `open` articulations. See [\[undefined\]](#) [\[Articulations and ornamentations\]](#), pagina [\[undefined\]](#), and [\[undefined\]](#) [\[List of articulations\]](#), pagina [\[undefined\]](#).
- Stopped horns are indicated by the `stopped` articulation. See [\[undefined\]](#) [\[Articulations and ornamentations\]](#), pagina [\[undefined\]](#).

## Frammenti di codice selezionati

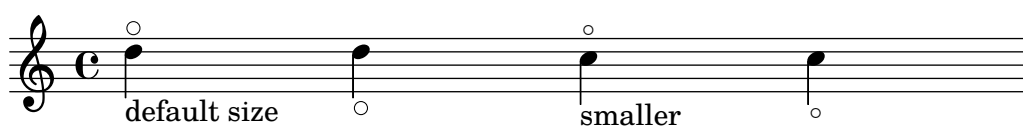
*Changing \flageolet mark size*

To make the `\flageolet` circle smaller use the following Scheme function.

```
smallFlageolet =
#(let ((m (make-articulation "flageolet")))
  (set! (ly:music-property m 'tweaks)
    (acons 'font-size -3
      (ly:music-property m 'tweaks)))
  m)

\layout { ragged-right = ##f }

\relative c'' {
  d4^\flageolet\markup { default size } d_\flageolet
  c4^\smallFlageolet\markup { smaller } c_\smallFlageolet
}
```



## Vedi anche

Notation Reference: [\[Breath marks\]](#), pagina [\[undefined\]](#), [\[Slurs\]](#), pagina [\[undefined\]](#), [\[Articulations and ornamentations\]](#), pagina [\[undefined\]](#), [\[List of articulations\]](#), pagina [\[undefined\]](#), [\[Tremolo repeats\]](#), pagina [\[undefined\]](#), [\[Instrument transpositions\]](#), pagina [\[undefined\]](#), [\[Glissando\]](#), pagina 140, [\[Grace notes\]](#), pagina [\[undefined\]](#), [\[Falls and doits\]](#), pagina [\[undefined\]](#), [\[Special note heads\]](#), pagina [\[undefined\]](#),

Snippets: Sezione “Winds” in *Frammenti di codice*.

## Fingerings

All wind instruments other than the trombone require the use of several fingers to produce each pitch. Some fingering examples are shown in the snippets below.

Woodwind diagrams can be produced and are described in Sezione 2.6.3.1 [\[Woodwind diagrams\]](#), pagina 402.

## Frammenti di codice selezionati

### *Fingering symbols for wind instruments*

Special symbols can be achieved by combining existing glyphs, which is useful for wind instruments.

```
centermarkup = {
  \once \override TextScript.self-alignment-X = #CENTER
  \once \override TextScript.X-offset =#(lambda (g)
    (+ (ly:self-alignment-interface::centered-on-x-parent g)
      (ly:self-alignment-interface::x-aligned-on-self g)))
}

\score {
  \relative c'{
    g\open
    \once \override TextScript.staff-padding = #-1.0
    \centermarkup
    g^\markup {
      \combine
      \musicglyph #"scripts.open"
      \musicglyph #"scripts.tenuto"
    }
    \centermarkup
    g^\markup {
      \combine
      \musicglyph #"scripts.open"
      \musicglyph #"scripts.stopped"
    }
    g\stopped
  }
}
```



### *Recorder fingering chart*

The following example demonstrates how fingering charts for wind instruments can be realized.

% range chart for paetzold contrabass recorder

```
centermarkup = {
  \once \override TextScript.self-alignment-X = #CENTER
  \once \override TextScript.X-offset =#(lambda (g)
    (+ (ly:self-alignment-interface::centered-on-x-parent g)
      (ly:self-alignment-interface::x-aligned-on-self g)))
}

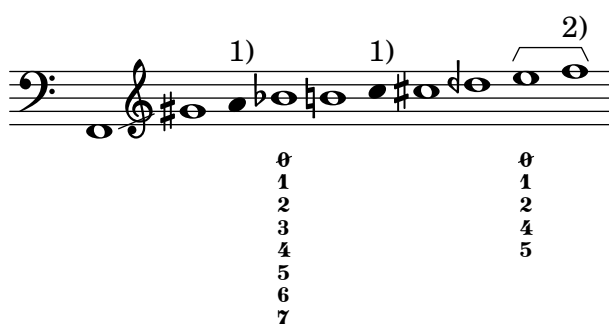
\score {
  \new Staff \with {
    \remove "Time_signature_engraver"
```



```

\omit Stem
\omit Flag
\consists "Horizontal_bracket_engraver"
}
{
\clef bass
\set Score.timing = ##f
f,1*1/4 \glissando
\clef violin
gis'1*1/4
\stemDown a'4^\markup{1)}
\centermarkup
\once \override TextScript.padding = #2
bes'1*1/4_\markup{\override #'(baseline-skip . 1.7) \column
  { \fontsize #-5 \slashed-digit #0 \finger 1 \finger 2 \finger 3 \finger 4
\finger 5 \finger 6 \finger 7} }
b'1*1/4
c''4^\markup{1)}
\centermarkup
\once \override TextScript.padding = #2
cis''1*1/4
deh''1*1/4
\centermarkup
\once \override TextScript.padding = #2
\once \override Staff.HorizontalBracket.direction = #UP
e''1*1/4_\markup{\override #'(baseline-skip . 1.7) \column
  { \fontsize #-5 \slashed-digit #0 \finger 1 \finger 2 \finger 4
\finger 5} }\startGroup
f''1*1/4^\markup{2)}\stopGroup
}
}

```



## Vedi anche

Notation Reference: Sezione 2.6.3.1 [Woodwind diagrams], pagina 402.

Snippets: Sezione “Winds” in *Frammenti di codice*.

## 2.6.2 Bagpipes

This section discusses notation common bagpipes.

### Bagpipe definitions

LilyPond contains special definitions for Scottish, Highland Bagpipe music; to use them, add

```
\include "bagpipe.ly"
```

to the top of your input file. This lets you add the special grace notes common to bagpipe music with short commands. For example, you could write `\taor` instead of

```
\grace { \small G32[ d G e] }
```

`bagpipe.ly` also contains pitch definitions for the bagpipe notes in the appropriate octaves, so you do not need to worry about `\relative` or `\transpose`.

```
\include "bagpipe.ly"
```

```
{ \grg G4 \grg a \grg b \grg c \grg d \grg e \grg f \grA g A }
```



Bagpipe music nominally uses the key of D Major (even though that isn't really true). However, since that is the only key that can be used, the key signature is normally not written out. To set this up correctly, always start your music with `\hideKeySignature`. If you for some reason want to show the key signature, you can use `\showKeySignature` instead.

Some modern music use cross fingering on c and f to flatten those notes. This can be indicated by c-flat or f-flat. Similarly, the piobaireachd high g can be written g-flat when it occurs in light music.

## Vedi anche

Snippets: Sezione “Winds” in *Frammenti di codice*.

## Bagpipe example

This is what the well known tune Amazing Grace looks like in bagpipe notation.

```
\include "bagpipe.ly"
```

```
\layout {
```

```
  indent = 0.0\cm
```

```
  \context { \Score \remove "Bar_number_engraver" }
```

```
}
```

```
\header {
```

```
  title = "Amazing Grace"
```

```
  meter = "Hymn"
```

```
  arranger = "Trad. arr."
```

```
}
```

```
{
```

```
  \hideKeySignature
```

```
  \time 3/4
```

```
  \grg \partial 4 a8. d16
```

```
  \slurd d2 \grg f8[ e32 d16.]
```

```
  \grg f2 \grg f8 e
```

```
  \thrwd d2 \grg b4
```

```
  \grG a2 \grg a8. d16
```

```
  \slurd d2 \grg f8[ e32 d16.]
```

```
  \grg f2 \grg e8. f16
```

```
  \dblA A2 \grg A4
```

```

\grg A2 f8. A16
\grg A2 \hdbl f8[ e32 d16.]
\grg f2 \grg f8 e
\thrwd d2 \grg b4
\grG a2 \grg a8. d16
\slurd d2 \grg f8[ e32 d16.]
\grg f2 e4
\thrwd d2.
\slurd d2
\bar "|."
}

```

## Amazing Grace

Hymn

Trad. arr.



### Vedi anche

Snippets: Sezione “Winds” in *Frammenti di codice*.

### 2.6.3 Woodwinds

This section discusses notation specifically for woodwind instruments.

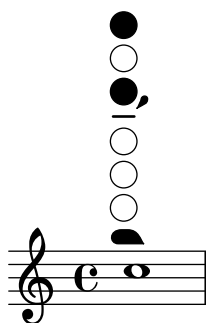
#### 2.6.3.1 Woodwind diagrams

Woodwind diagrams can be used to indicate the fingering to be used for specific notes and are available for the following instruments:

- piccolo
- flute
- oboe
- clarinet
- bass clarinet
- saxophone
- bassoon
- contrabassoon

Woodwind diagrams are created as markups:

```
c''1^\markup {
  \woodwind-diagram #'piccolo #'((lh . (gis))
                                (cc . (one three))
                                (rh . (ees)))
}
```



Keys can be open, partially-covered, ring-depressed, or fully covered:

```
\textLengthOn
c''1^\markup {
  \center-column {
    "one quarter"
    \woodwind-diagram #'flute #'((cc . (one1q))
                                  (lh . ())
                                  (rh . ()))
  }
}
```

```
c''1^\markup {
  \center-column {
    "one half"
    \woodwind-diagram #'flute #'((cc . (one1h))
                                  (lh . ())
                                  (rh . ()))
  }
}
```

```
c''1^\markup {
  \center-column {
    "three quarter"
    \woodwind-diagram #'flute #'((cc . (one3q))
                                  (lh . ())
                                  (rh . ()))
  }
}
```

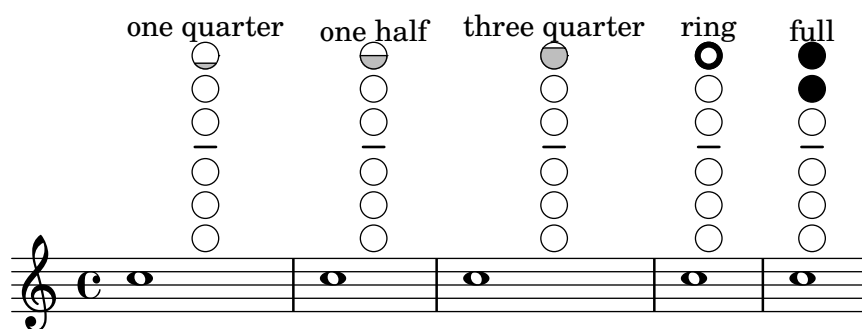
```
c''1^\markup {
  \center-column {
    "ring"
    \woodwind-diagram #'flute #'((cc . (oneR))
                                  (lh . ()))
  }
}
```

```

                                (rh . ()))
}
}

c''1^\markup {
  \center-column {
    "full"
    \woodwind-diagram #'flute #'((cc . (oneF two))
                                (lh . ()))
                                (rh . ()))
  }
}

```

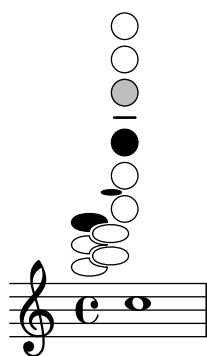


Trills are indicated as shaded keys:

```

c''1^\markup {
  \woodwind-diagram #'bass-clarinete
    #'((cc . (threeT four))
        (lh . ()))
        (rh . (b fis)))
}

```



A variety of trills can be displayed:

```

\textLengthOn
c''1^\markup {
  \center-column {
    "one quarter to ring"
    \woodwind-diagram #'flute #'((cc . (one1qTR))
                                (lh . ()))
                                (rh . ()))
  }
}

```

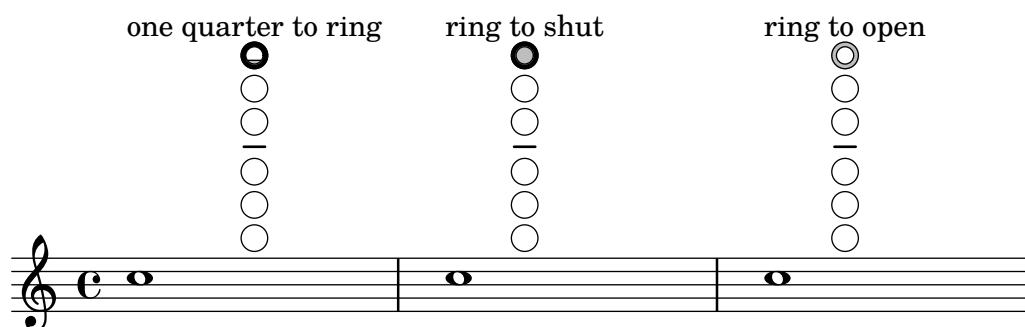
}

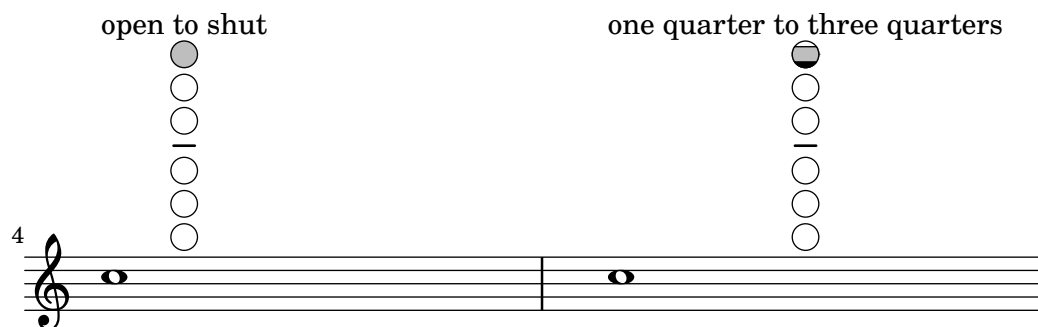
```
c''1^\markup {
  \center-column {
    "ring to shut"
    \woodwind-diagram #'flute #'((cc . (oneTR))
                        (lh . ()))
                        (rh . ()))
  }
}
```

```
c''1^\markup {
  \center-column {
    "ring to open"
    \woodwind-diagram #'flute #'((cc . (oneRT))
                        (lh . ()))
                        (rh . ()))
  }
}
```

```
c''1^\markup {
  \center-column {
    "open to shut"
    \woodwind-diagram #'flute #'((cc . (oneT))
                        (lh . ()))
                        (rh . ()))
  }
}
```

```
c''1^\markup {
  \center-column {
    "one quarter to three quarters"
    \woodwind-diagram #'flute #'((cc . (one1qT3q))
                        (lh . ()))
                        (rh . ()))
  }
}
```





The list of all possible keys and settings for a given instrument can be displayed on the console using `#(print-keys-verbose 'flute)` or in the log file using `#(print-keys-verbose 'flute (current-error-port))`, although they will not show up in the music output.

Creating new diagrams is possible, although this will require Scheme ability and may not be accessible to all users. The patterns for the diagrams are in `scm/define-woodwind-diagrams.scm` and `scm/display-woodwind-diagrams.scm`.

## Comandi predefiniti

### Frammenti di codice selezionati

*Woodwind diagrams listing*

The following music shows all of the woodwind diagrams currently defined in LilyPond.

```
\layout {
  indent = 0
}

\relative c' {
  \textLengthOn
  c1^
  \markup {
    \center-column {
      'tin-whistle
      " "
      \woodwind-diagram
      #'tin-whistle
      #'()
    }
  }

  c1^
  \markup {
    \center-column {
      'piccolo
      " "
      \woodwind-diagram
      #'piccolo
      #'()
    }
  }

  c1^
  \markup {
    \center-column {
      'flute
```

```

        " "
        \woodwind-diagram
        #'flute
        #'()
    }
}
c1^\markup {
  \center-column {
    'oboe
    " "
    \woodwind-diagram
    #'oboe
    #'()
  }
}

c1^\markup {
  \center-column {
    'clarinet
    " "
    \woodwind-diagram
    #'clarinet
    #'()
  }
}

c1^\markup {
  \center-column {
    'bass-clarinet
    " "
    \woodwind-diagram
    #'bass-clarinet
    #'()
  }
}

c1^\markup {
  \center-column {
    'saxophone
    " "
    \woodwind-diagram
    #'saxophone
    #'()
  }
}

c1^\markup {
  \center-column {
    'bassoon
    " "
    \woodwind-diagram
    #'bassoon
  }
}

```

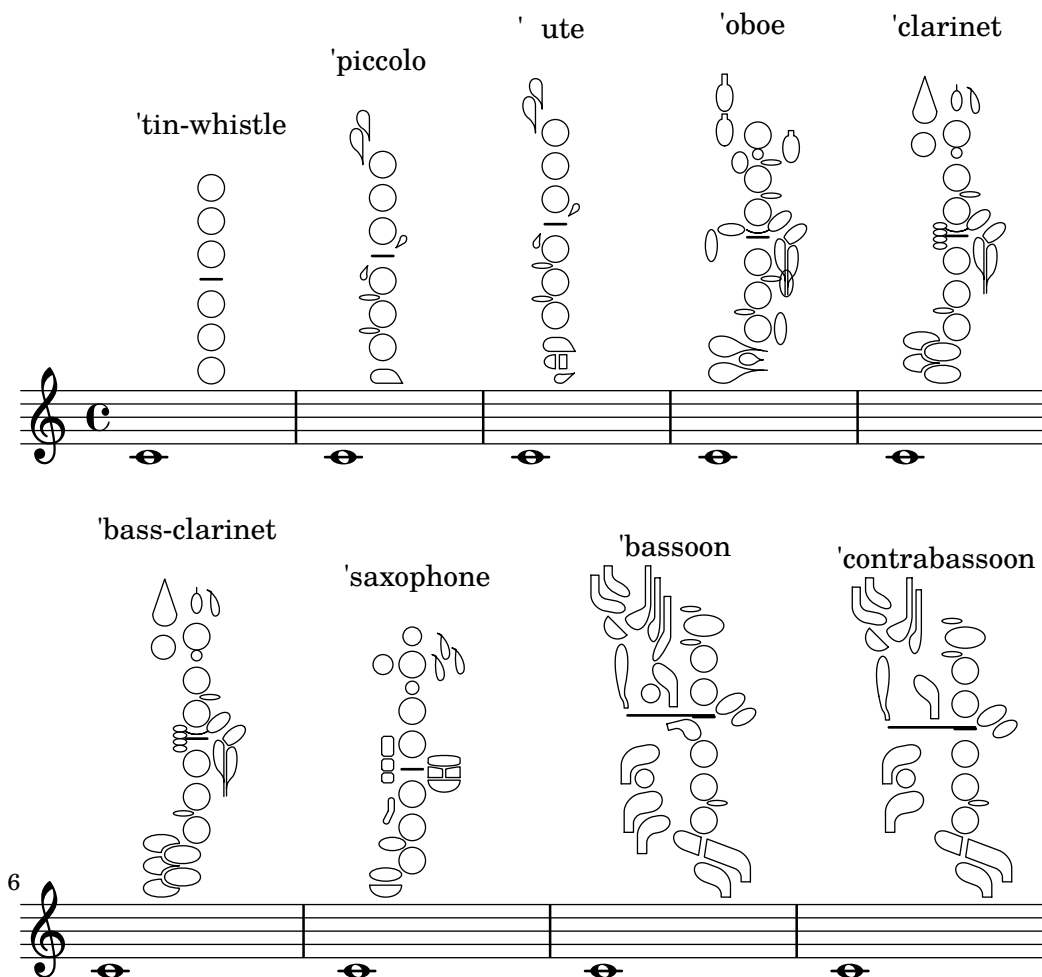


```

      #'()
    }
  }

c1^\markup {
  \center-column {
    'contrabassoon
    " "
    \woodwind-diagram
    #'contrabassoon
    #'()
  }
}

```



### *Graphical and text woodwind diagrams*

In many cases, the keys other than the central column can be displayed by key name as well as by graphical means.

```

\relative c'' {
  \textLengthOn
  c1^\markup
  \woodwind-diagram
  #'piccolo
}

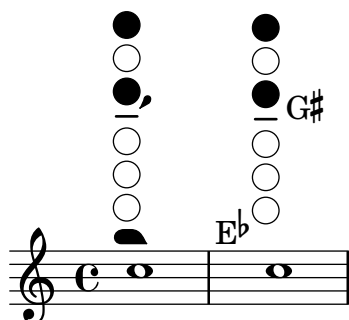
```

```

#'((cc . (one three))
    (lh . (gis))
    (rh . (ees)))

c^\markup
  \override #'(graphical . #f) {
    \woodwind-diagram
      #'piccolo
      #'((cc . (one three))
          (lh . (gis))
          (rh . (ees)))
  }
}

```



### *Changing the size of woodwind diagrams*

The size and thickness of woodwind diagrams can be changed.

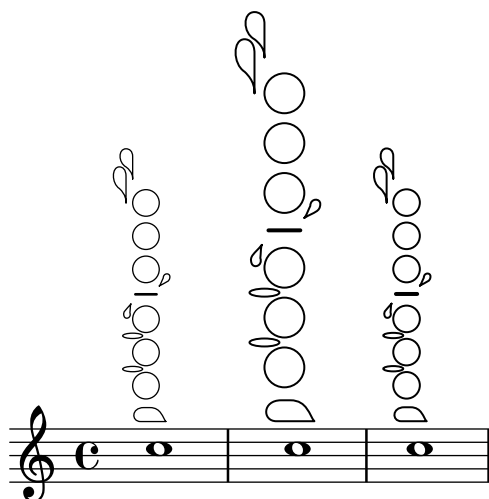
```

\relative c'' {
  \textLengthOn
  c1^\markup
    \woodwind-diagram
      #'piccolo
      #'()

  c^\markup
    \override #'(size . 1.5) {
      \woodwind-diagram
        #'piccolo
        #'()
    }

  c^\markup
    \override #'(thickness . 0.15) {
      \woodwind-diagram
        #'piccolo
        #'()
    }
}

```



### *Woodwind diagrams key lists*

The snippet below produces a list of all possible keys and key settings for woodwind diagrams as defined in `scm/define-woodwind-diagrams.scm`. The list will be displayed in the log file, but not in the music. If output to the console is wanted, omit the `(current-error-port)` from the commands.

```
#(print-keys-verbose 'piccolo (current-error-port))
#(print-keys-verbose 'flute (current-error-port))
#(print-keys-verbose 'flute-b-extension (current-error-port))
#(print-keys-verbose 'tin-whistle (current-error-port))
#(print-keys-verbose 'oboe (current-error-port))
#(print-keys-verbose 'clarinet (current-error-port))
#(print-keys-verbose 'bass-clarinet (current-error-port))
#(print-keys-verbose 'low-bass-clarinet (current-error-port))
#(print-keys-verbose 'saxophone (current-error-port))
#(print-keys-verbose 'soprano-saxophone (current-error-port))
#(print-keys-verbose 'alto-saxophone (current-error-port))
#(print-keys-verbose 'tenor-saxophone (current-error-port))
#(print-keys-verbose 'baritone-saxophone (current-error-port))
#(print-keys-verbose 'bassoon (current-error-port))
#(print-keys-verbose 'contrabassoon (current-error-port))
```

```
\score {c'1}
```



### **Vedi anche**

Installed Files: `scm/define-woodwind-diagrams.scm`,  
`scm/display-woodwind-diagrams.scm`.

Snippets: Sezione “Winds” in *Frammenti di codice*.

Internals Reference: Sezione “TextScript” in *Guida al Funzionamento Interno*, Sezione “instrument-specific-markup-interface” in *Guida al Funzionamento Interno*.

## 2.7 Chord notation

Chord notation for the first system:

F C F F C F

1. Fair is the sun - shine, Fair - er the moon - light  
 2. Fair are the mead - ows, Fair - er the wood - land,

Chord notation for the second system:

F B $\flat$  F C<sup>7</sup> F C

And all the stars in heav'n a - bove;  
 Robed in the ow - ers of bloom - ing spring;

Chords can be entered either as normal notes or in chord mode and displayed using a variety of traditional European chord naming conventions. Chord names and figured bass notation can also be displayed.

### 2.7.1 Chord mode

Chord mode is used to enter chords using an indicator of the chord structure, rather than the chord pitches.

#### Chord mode overview

Chords can be entered as simultaneous music, as discussed in [\[Chorded notes\]](#), pagina [\[undefined\]](#).

Chords can also be entered in “chord mode”, which is an input mode that focuses on the structures of chords in traditional European music, rather than on specific pitches. This is convenient for those who are familiar with using chord names to describe chords. More information on different input modes can be found at Sezione 5.4.1 [\[Input modes\]](#), pagina 610.

```
\chordmode { c1 g a g c }
```

Chords entered using chord mode are music elements, and can be transposed just like chords entered using simultaneous music. `\chordmode` is absolute, as `\relative` has no effect on `chordmode` blocks. However, in `\chordmode` the absolute pitches are one octave higher than in note mode.

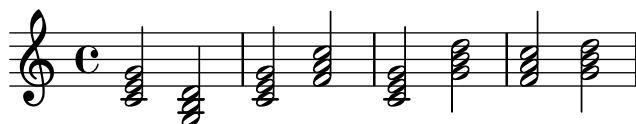
Chord mode and note mode can be mixed in sequential music:

```
\relative {
```

```

<c' e g>2 <g b d>
\chordmode { c2 f }
<c e g>2 <g' b d>
\chordmode { f2 g }
}

```



## Vedi anche

Music Glossary: Sezione “chord” in *Glossario Musicale*.

Notation Reference: [\[Chorded notes\]](#), pagina [\[undefined\]](#), Sezione 5.4.1 [\[Input modes\]](#), pagina 610.

Snippets: Sezione “Chords” in *Frammenti di codice*.

## Problemi noti e avvertimenti

Predefined shorthands for articulations and ornaments cannot be used on notes in chord mode, see [\[Articulations and ornamentations\]](#), pagina [\[undefined\]](#).

## Common chords

Major triads are entered by including the root and an optional duration:

```
\chordmode { c2 f4 g }
```



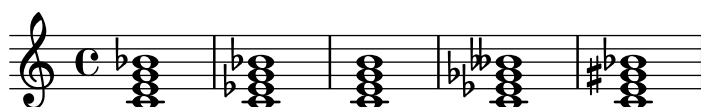
Minor, augmented, and diminished triads are entered by placing : and a quality modifier string after the duration:

```
\chordmode { c2:m f4:aug g:dim }
```

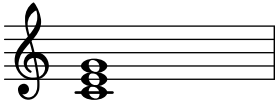


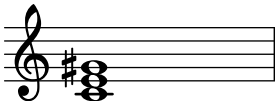
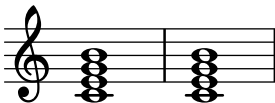


Seventh chords can be created:

```
\chordmode { c1:7 c:m7 c:maj7 c:dim7 c:aug7 }
```



The table below shows the actions of the quality modifiers on triads and seventh chords. The default seventh step added to chords is a minor or flatted seventh, which makes the dominant seventh the basic seventh chord. All alterations are relative to the dominant seventh. A more complete table of modifier usage is found at [\[Common chord modifiers\]](#), pagina [\[undefined\]](#).

Modifier	Action	Example
None	The default action; produces a major triad.	
m, m7	The minor chord. This modifier lowers the 3rd.	
dim, dim7	The diminished chord. This modifier lowers the 3rd, 5th and (if present) the 7th step.	
aug	The augmented chord. This modifier raises the 5th step.	
maj, maj7	The major 7th chord. This modifier adds a raised 7th step. The 7 following maj is optional. Do NOT use this modifier to create a major triad.	

## Vedi anche

Notation Reference: [\[Common chord modifiers\]](#), pagina [\[Extended and altered chords\]](#), pagina 413.

Snippets: Sezione “Chords” in *Frammenti di codice*.

## Problemi noti e avvertimenti

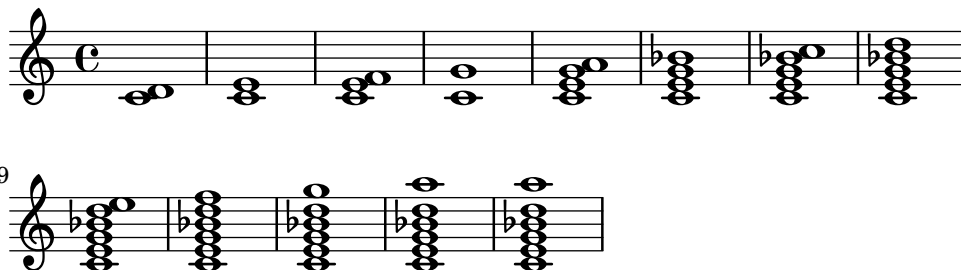
Only one quality modifier should be used per chord, typically on the highest step present in the chord. Chords with more than quality modifier will be parsed without an error or warning, but the results are unpredictable. Chords that cannot be achieved with a single quality modifier should be altered by individual pitches, as described in [\[Extended and altered chords\]](#), pagina 413.

## Extended and altered chords

Chord structures of arbitrary complexity can be created in chord mode. The modifier string can be used to extend a chord, add or remove chord steps, raise or lower chord steps, and add a bass note or create an inversion.

The first number following the `:` is taken to be the extent of the chord. The chord is constructed by sequentially adding thirds to the root until the specified number has been reached. Note that the seventh step added as part of an extended chord will be the minor or flatted seventh, not the major seventh. If the extent is not a third (e.g., 6), thirds are added up to the highest third below the extent, and then the step of the extent is added. The largest possible value for the extent is 13. Any larger value is interpreted as 13.

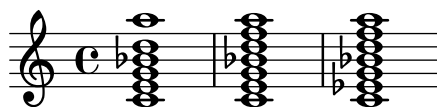
```
\chordmode {
  c1:2 c:3 c:4 c:5
  c1:6 c:7 c:8 c:9
  c1:10 c:11 c:12 c:13
  c1:14
}
```



As a special exception, `c:5` produces a ‘power chord’ only consisting of root and fifth.

Since an unaltered 11 does not sound good when combined with an unaltered 13, the 11 is removed from a `:13` chord (unless it is added explicitly).

```
\chordmode {
  c1:13 c:13.11 c:m13
}
```



Individual steps can be added to a chord. Additions follow the extent and are prefixed by a dot (.). The basic seventh step added to a chord is the minor or flatted seventh, rather than the major seventh.

```
\chordmode {
  c1:3.5.6 c:3.7.8 c:3.6.13
}
```



Added steps can be as high as desired.

```
\chordmode {
  c4:3.5.15 c:3.5.20 c:3.5.25 c:3.5.30
}
```



Added chord steps can be altered by suffixing a - or + sign to the number. To alter a step that is automatically included as part of the basic chord structure, add it as an altered step.

```
\chordmode {
  c1:7+ c:5+.3- c:3-.5-.7-
}
```



Following any steps to be added, a series of steps to be removed is introduced in a modifier string with a prefix of ^. If more than one step is to be removed, the steps to be removed are separated by . following the initial ^.

```
\chordmode {
  c1^3 c:7^5 c:9^3 c:9^3.5 c:13.11^3.7
}
```



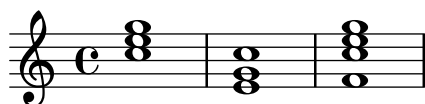
The modifier *sus* can be added to the modifier string to create suspended chords. This removes the 3rd step from the chord. Append either 2 or 4 to add the 2nd or 4th step to the chord. When *sus* is followed by either a 2nd or 4th step, it is equivalent to ^3, otherwise to *sus4*, namely 5.4.

```
\chordmode {
  c1:sus c:sus2 c:sus4 c:5.4
}
```



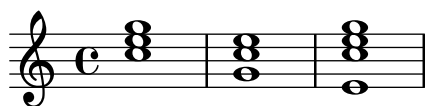
Inversions (putting a pitch other than the root on the bottom of the chord) and added bass notes can be specified by appending */pitch* to the chord.

```
\chordmode {
  c'1 c'/e c'/f
}
```



A bass note that is part of the chord can be added, instead of moved as part of an inversion, by using */+pitch*.

```
\chordmode {
  c'1 c'/g c'/+e
}
```





Chord modifiers that can be used to produce a variety of standard chords are shown in [\[Common chord modifiers\]](#), pagina [\[undefined\]](#).

## Vedi anche

Notation Reference: [\[Common chord modifiers\]](#), pagina [\[undefined\]](#).

Snippets: Sezione “Chords” in *Frammenti di codice*.

## Problemi noti e avvertimenti

Each step can only be present in a chord once. The following simply produces the augmented chord, since 5+ is interpreted last.

```
\chordmode { c1:3.5.5-.5+ }
```



### 2.7.2 Displaying chords

Chords can be displayed by name, in addition to the standard display as notes on a staff.

## Printing chord names

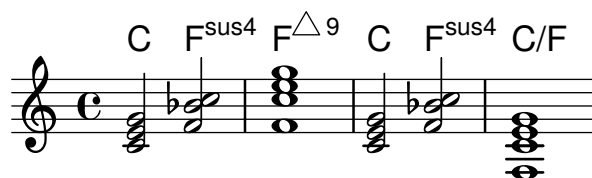
Chord names are printed in the `ChordNames` context:

```
\new ChordNames {
  \chordmode {
    c2 f4. g8
  }
}
```

C F G

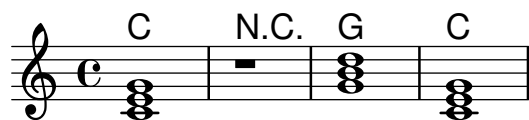
Chords can be entered as simultaneous notes or through the use of chord mode. The displayed chord name will be the same, regardless of the mode of entry, unless there are inversions or added bass notes:

```
chordmusic = \relative {
  <c' e g>2 <f bes c>
  <f c' e g>1
  \chordmode {
    c2 f:sus4 c1:/f
  }
}
<<
  \new ChordNames {
    \chordmusic
  }
  {
    \chordmusic
  }
>>
```



Rests passed to a `ChordNames` context will cause the `noChordSymbol` markup to be displayed.

```
<<
\new ChordNames \chordmode {
  c1
  r1
  g1
  c1
}
\chordmode {
  c1
  r1
  g1
  c1
}
>>
```



`\chords { ... }` is a shortcut notation for `\new ChordNames { \chordmode { ... } }`.

```
\chords {
  c2 f4.:m g8:maj7
}
```

C Fm G<sup>Δ</sup>

```
\new ChordNames {
  \chordmode {
    c2 f4.:m g8:maj7
  }
}
```

C Fm G<sup>Δ</sup>

## Frammenti di codice selezionati

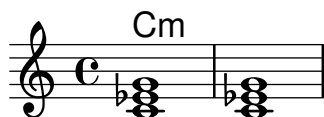
*Showing chords at changes*

Chord names can be displayed only at the start of lines and when the chord changes.

```
harmonies = \chordmode {
  c1:m c:m \break c:m c:m d
}
```

```
<<
\new ChordNames {
  \set chordChanges = ##t
  \harmonies
}
```

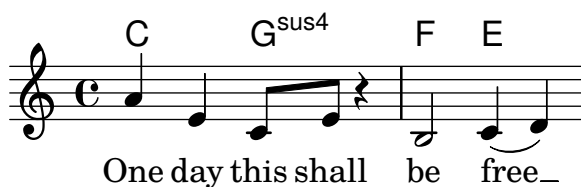
```
\new Staff {
  \relative c' { \harmonies }
}
>>
```



### *Canzoniere semplice*

Mettendo insieme nomi degli accordi, melodia e testo si ottiene un canzoniere (in inglese “lead sheet”):

```
<<
\chords { c2 g:sus4 f e }
\relative c'' {
  a4 e c8 e r4
  b2 c4( d)
}
\addlyrics { One day this shall be free __ }
>>
```



## Vedi anche

Music Glossary: Sezione “chord” in *Glossario Musicale*.

Notation Reference: [\[Writing music in parallel\]](#), pagina [\[undefined\]](#).

Snippets: Sezione “Chords” in *Frammenti di codice*.

Internals Reference: Sezione “ChordNames” in *Guida al Funzionamento Interno*, Sezione “ChordName” in *Guida al Funzionamento Interno*, Sezione “Chord\_name\_engraver” in *Guida al Funzionamento Interno*, Sezione “Volta\_engraver” in *Guida al Funzionamento Interno*, Sezione “Bar\_engraver” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Chords containing inversions or altered bass notes are not named properly if entered using simultaneous music.


## Customizing chord names

There is no unique system for naming chords. Different musical traditions use different names for the same set of chords. There are also different symbols displayed for a given chord name. The names and symbols displayed for chord names are customizable.

The basic chord name layout is a system for Jazz music, proposed by Klaus Ignatzek (see Sezione “Literature list” in *Saggio*). The chord naming system can be modified as described below. An alternate jazz chord system has been developed using these modifications. The Ignatzek and alternate Jazz notation are shown on the chart in [Chord name chart], pagina (undefined).

In addition to the different naming systems, different note names are used for the root in different languages. The predefined commands `\germanChords`, `\semiGermanChords`, `\italianChords` and `\frenchChords` set these variables. The effect is demonstrated here:

default	E/D	Cm	B/B	B <sup>#</sup> /B <sup>#</sup>	B <sup>b</sup> /B <sup>b</sup>
german	E/d	Cm	H/h	H <sup>#</sup> /his	B/b
semi-german	E/d	Cm	H/h	H <sup>#</sup> /his	B <sup>b</sup> /b
italian	Mi/Re	Do m	Si/Si	Si <sup>#</sup> /Si <sup>#</sup>	Si <sup>b</sup> /Si <sup>b</sup>
french	Mi/Ré	Do m	Si/Si	Si <sup>#</sup> /Si <sup>#</sup>	Si <sup>b</sup> /Si <sup>b</sup>



German songbooks may indicate minor chords as lowercase letters, without any *m* suffix. This can be obtained by setting the `chordNameLowercaseMinor` property:

```
\chords {
  \set chordNameLowercaseMinor = ##t
  c2 d:m e:m f
}
```

C d e F

If none of the existing settings give the desired output, the chord name display can be tuned through the following properties.

#### chordRootNamer

The chord name is usually printed as a letter for the root with an optional alteration. The transformation from pitch to letter is done by this function. Special note names (for example, the German ‘H’ for a B-chord) can be produced by storing a new function in this property.

#### majorSevenSymbol

This property contains the markup object used to follow the output of `chordRootNamer` to identify a major 7 chord. Predefined options are `whiteTriangleMarkup` and `blackTriangleMarkup`.

#### additionalPitchPrefix

When the chord name contains additional pitches, they can optionally be prefixed with some text. The default is no prefix, in order to avoid too much visual clutter, but for small numbers of additional pitches this can be visually effective.

```
\new ChordNames {
  <c e g d'> % add9
  \set additionalPitchPrefix = #"add"
  <c e g d'> % add9
}
```

$C^9$   $C^{add9}$

#### `chordNoteNamer`

When the chord name contains additional pitches other than the root (e.g., an added bass note), this function is used to print the additional pitch. By default the pitch is printed using `chordRootNamer`. The `chordNoteNamer` property can be set to a specialized function to change this behavior. For example, the bass note can be printed in lower case.

#### `chordNameSeparator`

Different parts of a chord name are normally separated by a small amount of horizontal space. By setting `chordNameSeparator`, you can use any desired markup for a separator. This does not affect the separator between a chord and its bass note; to customize that, use `slashChordSeparator`.

```
\chords {
  c4:7.9- c:7.9-/g
  \set chordNameSeparator = \markup { "/" }
  \break
  c4:7.9- c:7.9-/g
}
```

$C^{7\flat9}$   $C^{7\flat9}/G$

$C^{7/\flat9}$   $C^{7/\flat9}/G$

#### `slashChordSeparator`

Chords can be played over a bass note other than the conventional root of the chord. These are known as “inversions” or “slash chords”, because the default way of notating them is with a forward slash between the main chord and the bass note. Therefore the value of `slashChordSeparator` defaults to a forward slash, but you can change it to any markup you choose.

```
\chords {
  c4:7.9- c:7.9-/g
  \set slashChordSeparator = \markup { " over " }
  \break
  c4:7.9- c:7.9-/g
}
```

$C^{7\flat9}$   $C^{7\flat9}/G$

$C^{7\flat9}$   $C^{7\flat9}$  over G

#### `chordNameExceptions`

This property is a list of pairs. The first item in each pair is a set of pitches used to identify the steps present in the chord. The second item is a markup that will follow the `chordRootNamer` output to create the chord name.

#### `minorChordModifier`

Minor chords are often denoted via a ‘m’ suffix to the right of the root of the chord. However some idioms prefer other suffices, such as a minus sign.

```
\chords {
```

```

c4:min f:min7
\set minorChordModifier = \markup { "-" }
\break
c4:min f:min7
}

```

Cm Fm<sup>7</sup>

C- F-<sup>7</sup>

#### chordPrefixSpacer

The modifier for minor chords as determined by `minorChordModifier` is usually printed immediately to the right of the root of the chord. A spacer can be placed between the root and the modifier by setting `chordPrefixSpacer`. The spacer is not used when the root is altered.

## Comandi predefiniti

`\whiteTriangleMarkup`, `\blackTriangleMarkup`, `\germanChords`, `\semiGermanChords`, `\italianChords`, `\frenchChords`.

## Frammenti di codice selezionati

### *Chord name exceptions*

The property `chordNameExceptions` can be used to store a list of special notations for specific chords.

```

% modify maj9 and 6(add9)
% Exception music is chords with markups
chExceptionMusic = {
  <c e g b d'>1-\markup { \super "maj9" }
  <c e g a d'>1-\markup { \super "6(add9)" }
}

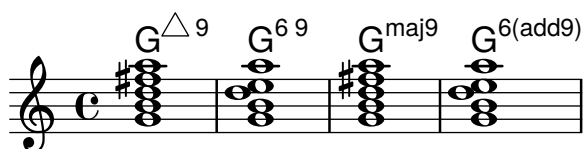
% Convert music to list and prepend to existing exceptions.
chExceptions = #( append
  ( sequential-music-to-chord-exceptions chExceptionMusic #t)
  ignatzekExceptions)

theMusic = \chordmode {
  g1:maj9 g1:6.9
  \set chordNameExceptions = #chExceptions
  g1:maj9 g1:6.9
}

\layout {
  ragged-right = ##t
}

<< \context ChordNames \theMusic
  \context Voice \theMusic
>>

```



*chord name major7*

The layout of the major 7 can be tuned with `majorSevenSymbol`.

```
\chords {
  c:7+
  \set majorSevenSymbol = \markup { j7 }
  c:7+
}
```

$C^{\Delta} C^{j7}$

*Adding bar lines to ChordNames context*

To add bar line indications in the `ChordNames` context, add the `Bar_engraver`.

```
\new ChordNames \with {
  \override BarLine.bar-extent = #'(-2 . 2)
  \consists "Bar_engraver"
}
```

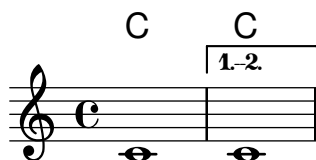
```
\chordmode {
  f1:maj7 f:7 bes:7
}
```

$F^{\Delta} \mid F^7 \mid B\flat^7 \mid$

*Volta below chords*

By adding the `Volta_engraver` to the relevant staff, volte can be put under chords.

```
\score {
  <<
    \chords {
      c1
      c1
    }
    \new Staff \with {
      \consists "Volta_engraver"
    }
    {
      \repeat volta 2 { c'1 }
      \alternative { c' }
    }
  >>
  \layout {
    \context {
      \Score
      \remove "Volta_engraver"
    }
  }
}
```



### Changing chord separator

The separator between different parts of a chord name can be set to any markup.

```
\chords {
  c:7sus4
  \set chordNameSeparator
    = \markup { \typewriter | }
  c:7sus4
}
```

$C^7 \text{ sus4 } C^7 | \text{ sus4}$

### Vedi anche

Notation Reference: [\[Chord name chart\]](#), pagina [\[Common chord modifiers\]](#), pagina [\[Chord name chart\]](#), pagina [\[Common chord modifiers\]](#).

Essay on automated music engraving: Sezione “Literature list” in *Saggio*.

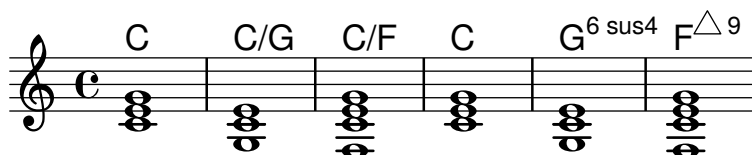
Installed Files: [scm/chords-ignatzek.scm](#), [scm/chord-entry.scm](#), [ly/chord-modifier-init.ly](#).

Snippets: Sezione “Chords” in *Frammenti di codice*.

### Problemi noti e avvertimenti

Chord names are determined from both the pitches that are present in the chord and the information on the chord structure that may have been entered in `\chordmode`. If the simultaneous pitches method of entering chords is used, undesired names result from inversions or bass notes.

```
myChords = \relative c' {
  \chordmode { c1 c/g c/f }
  <c e g>1 <g c e> <f c' e g>
}
<<
  \new ChordNames { \myChords }
  \new Staff { \myChords }
>>
```





### 2.7.3 Figured bass

*Adagio.*

Violino I.

Violino II.

Violone,  
e Cembalo.

6 # 6 6 6 6 #

6 4+ 2

5 6 6 5 5 6 6 5 #

6 # 6 6 5 4 6 6 5 4 3 7 6 5 9 8 4 3

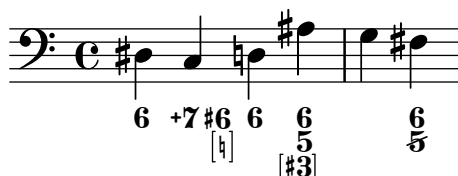
Figured bass notation can be displayed.

#### Introduction to figured bass

LilyPond has support for figured bass, also called thorough bass or basso continuo:

```
<<
\new Voice { \clef bass dis4 c d ais g fis}
\new FiguredBass {
  \figuremode {
    < 6 >4 < 7\+ >8 < 6+ [_!] >
    < 6 >4 < 6 5 [3+] >
    < _ >4 < 6 5/>4
  }
}
```

```
}
>>
```



The support for figured bass consists of two parts: there is an input mode, introduced by `\figuremode`, that accepts entry of bass figures, and there is a context named `FiguredBass` that takes care of displaying `BassFigure` objects. Figured bass can also be displayed in `Staff` contexts.

`\figures{ ... }` is a shortcut notation for `\new FiguredBass { \figuremode { ... } }`.

Although the support for figured bass may superficially resemble chord support, it is much simpler. `\figuremode` mode simply stores the figures and the `FiguredBass` context prints them as entered. There is no conversion to pitches.

## Vedi anche

Music Glossary: Sezione “figured bass” in *Glossario Musicale*.

Snippets: Sezione “Chords” in *Frammenti di codice*.

## Entering figured bass

`\figuremode` is used to switch the input mode to figure mode. More information on different input modes can be found at Sezione 5.4.1 [Input modes], pagina 610.

In figure mode, a group of bass figures is delimited by `<` and `>`. The duration is entered after the `>`.

```
\new FiguredBass {
  \figuremode {
    <6 4>2
  }
}
```

**6**  
**4**

Accidentals (including naturals) can be added to figures:

```
\figures {
  <7! 6+ 4-> <5++> <3-->
}
```

**7** **x5** **3**  
**#6**  
**b4**

Augmented and diminished steps can be indicated:

```
\figures {
  <6\+ 5/> <7/>
}
```

**+6** **7**  
**5**

A backward slash through a figure (typically used for raised sixth steps) can be created:

```
\figures {
  <6> <6\\>
}
```

**6 6̸**

Vertical spaces and brackets can be included in figures:

```
\figures {
  <[12 _!] 8 [6 4]>
}
```

**[12  
‡  
8  
[6  
4]**

Any text markup can be inserted as a figure:

```
\figures {
  <\markup { \tiny \number 6 \super (1) } 5>
}
```

**6<sup>(1)</sup>  
5**

Continuation lines can be used to indicate repeated figures:

```
<<
{
  \clef bass
  e4 d c b,
  e4 d c b,
}
\figures {
  \bassFigureExtendersOn
  <6 4>4 <6 3> <7 3> <7 3>
  \bassFigureExtendersOff
  <6 4>4 <6 3> <7 3> <7 3>
}
>>
```



In this case, the extender lines replace existing figures, unless the continuation lines have been explicitly terminated.

```
<<
\figures {
  \bassFigureExtendersOn
  <6 4>4 <6 4> <6\! 4\!> <6 4>
}
>>
```

```
{
  \clef bass
  d4 d c c
}
```

>>



The table below summarizes the figure modifiers available.

Modifier	Purpose	Example
----------	---------	---------

+, -, !	Accidentals	
---------	-------------	--

$\flat 7$   $\times 5$   $\sharp 3$   
 $\sharp 6$   
 $\flat 4$

\+, /	Augmented and diminished steps	
-------	--------------------------------	--

$+6$   $7$   
 $5$

\	Raised sixth step	
---	-------------------	--

$\flat 6$

\!	End of continuation line	
----	--------------------------	--



## Comandi predefiniti

\bassFigureExtendersOn, \bassFigureExtendersOff.

## Frammenti di codice selezionati

*Changing the positions of figured bass alterations*

Accidentals and plus signs can appear before or after the numbers, depending on the `figuredBassAlterationDirection` and `figuredBassPlusDirection` properties.

```
\figures {
  <6\+> <5+> <6 4-> r
  \set figuredBassAlterationDirection = #RIGHT
  <6\+> <5+> <6 4-> r
  \set figuredBassPlusDirection = #RIGHT
  <6\+> <5+> <6 4-> r
  \set figuredBassAlterationDirection = #LEFT
  <6\+> <5+> <6 4-> r
}
```

$+6$   $\sharp 5$   $\flat 6$      $+6$   $\sharp 5$   $\flat 6$      $\flat 6$   $\sharp 5$   $\flat 6$      $\flat 6$   $\sharp 5$   $\flat 6$

## Vedi anche

Snippets: Sezione “Chords” in *Frammenti di codice*.

Internals Reference: Sezione “BassFigure” in *Guida al Funzionamento Interno*, Sezione “BassFigureAlignment” in *Guida al Funzionamento Interno*, Sezione “BassFigureLine” in *Guida al Funzionamento Interno*, Sezione “BassFigureBracket” in *Guida al Funzionamento Interno*, Sezione “BassFigureContinuation” in *Guida al Funzionamento Interno*, Sezione “FiguredBass” in *Guida al Funzionamento Interno*.

## Displaying figured bass

Figured bass can be displayed using the **FiguredBass** context, or in most staff contexts.

When displayed in a **FiguredBass** context, the vertical location of the figures is independent of the notes on the staff.

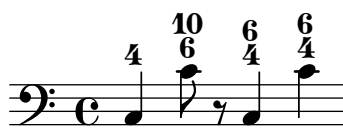
```
<<
  \relative {
    c''4 c'8 r8 c,4 c'
  }
  \new FiguredBass {
    \figuremode {
      <4>4 <10 6>8 s8
      <6 4>4 <6 4>
    }
  }
>>
```



In the example above, the **FiguredBass** context must be explicitly instantiated to avoid creating a second (empty) staff.

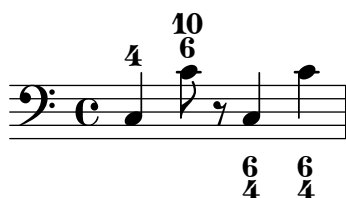
Figured bass can also be added to **Staff** contexts directly. In this case, the vertical position of the figures is adjusted automatically.

```
<<
  \new Staff = "myStaff"
  \figuremode {
    <4>4 <10 6>8 s8
    <6 4>4 <6 4>
  }
  %% Put notes on same Staff as figures
  \context Staff = "myStaff"
  {
    \clef bass
    c4 c'8 r8 c4 c'
  }
>>
```



When added in a `Staff` context, figured bass can be displayed above or below the staff.

```
<<
\new Staff = "myStaff"
\figuremode {
  <4>4 <10 6>8 s8
  \bassFigureStaffAlignmentDown
  <6 4>4 <6 4>
}
%% Put notes on same Staff as figures
\context Staff = "myStaff"
{
  \clef bass
  c4 c'8 r8 c4 c'
}
>>
```



## Comandi predefiniti

`\bassFigureStaffAlignmentDown`,  
`\bassFigureStaffAlignmentNeutral`.

`\bassFigureStaffAlignmentUp`,

## Vedi anche

Snippets: Sezione “Chords” in *Frammenti di codice*.

Internals Reference: Sezione “BassFigure” in *Guida al Funzionamento Interno*, Sezione “BassFigureAlignment” in *Guida al Funzionamento Interno*, Sezione “BassFigureLine” in *Guida al Funzionamento Interno*, Sezione “BassFigureBracket” in *Guida al Funzionamento Interno*, Sezione “BassFigureContinuation” in *Guida al Funzionamento Interno*, Sezione “FiguredBass” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

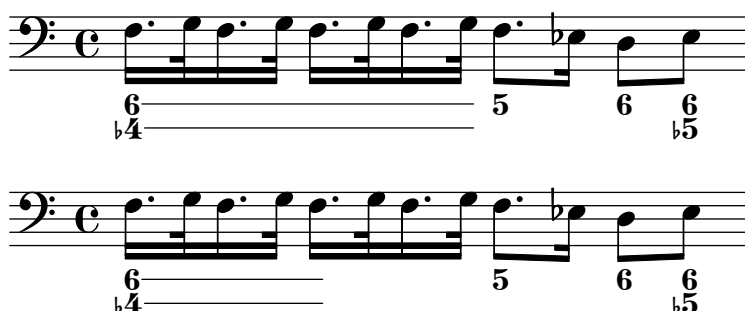
To ensure that continuation lines work properly, it is safest to use the same rhythm in the figure line as in the bass line.

```
<<
{
  \clef bass
  \repeat unfold 4 { f16. g32 } f8. es16 d8 es
}
\figures {
  \bassFigureExtendersOn
  % The extenders are correct here, with the same rhythm as the bass
  \repeat unfold 4 { <6 4->16. <6 4->32 }
  <5>8. r16 <6>8 <6\! 5->
}
>>
<<
```

```

{
  \clef bass
  \repeat unfold 4 { f16. g32 } f8. es16 d8 es
}
\figures {
  \bassFigureExtendersOn
  % The extenders are incorrect here, even though the timing is the same
  <6 4->4 <6 4->4
  <5>8. r16 <6>8 <6\! 5->
}
>>

```



## 2.8 Contemporary music

From the beginning of the 20th Century there has been a massive expansion of compositional style and technique. New harmonic and rhythmic developments, an expansion of the pitch spectrum and the development of a wide range of new instrumental techniques have been accompanied by a parallel evolution and expansion of musical notation. The purpose of this section is to provide references and information relevant to working with these new notational techniques.

### 2.8.1 Pitch and harmony in contemporary music

This section highlights issues that are relevant to notating pitch and harmony in contemporary music.

#### References for pitch and harmony in contemporary music

- Standard quarter-tone notation is addressed in [\[Note names in other languages\]](#), pagina [\[undefined\]](#).
- Non-standard key signatures are addressed in [\[Key signature\]](#), pagina [\[undefined\]](#).
- Contemporary practises in displaying accidentals are addressed in [\[Automatic accidentals\]](#), pagina [\[undefined\]](#).

#### Microtonal notation

#### Contemporary key signatures and harmony

### 2.8.2 Contemporary approaches to rhythm

This section highlights issues that are relevant to the notation of rhythm in contemporary music.

#### References for contemporary approaches to rhythm

- Compound time signatures are addressed in [\[Time signature\]](#), pagina [\[undefined\]](#).

- Basic polymetric notation is addressed in [\[Polymetric notation\]](#), pagina [\[undefined\]](#).
- Feathered beams are addressed in [\[Feathered beams\]](#), pagina [\[undefined\]](#).
- Mensurstriche bar lines (bar lines between staves only) are addressed in [\[Grouping staves\]](#), pagina [\[undefined\]](#).

## Tuplets in contemporary music

## Contemporary time signatures

## Extended polymetric notation

## Beams in contemporary music

## Bar lines in contemporary music

### 2.8.3 Graphical notation

### 2.8.4 Contemporary scoring techniques

### 2.8.5 New instrumental techniques

### 2.8.6 Further reading and scores of interest

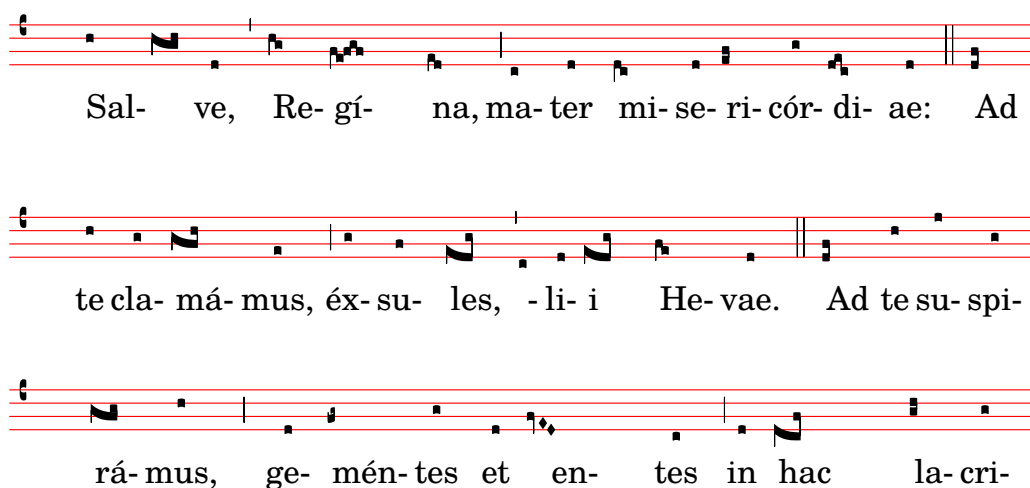
This section suggests books, musical examples and other resources useful in studying contemporary musical notation.

#### Books and articles on contemporary musical notation

- *Music Notation in the Twentieth Century: A Practical Guidebook* by Kurt Stone [W. W. Norton, 1980]
- *Music Notation: A Manual of Modern Practice* by Gardner Read [Taplinger, 1979]
- *Instrumentation and Orchestration* by Alfred Blatter [Schirmer, 2nd ed. 1997]

## Scores and musical examples

### 2.9 Ancient notation



Sal- ve, Re- gí- na, ma- ter mi- se- ri- cór- di- ae: Ad

te cla- má- mus, éx- su- les, - li- i He- vae. Ad te su- spi-

rá- mus, ge- mén- tes et en- tes in hac la- cri-





má-rum val- le. E-ia er-go, Ad-vo-cá- ta no-stra, il-

los tu- os mi-se-ri- cór- des ó-cu- los ad nos con- vér-te.

Et Je- sum, be- ne- díc- tum fruc- tum ven- tris tu- i, no-

bis post hoc ex- sí- li- um os- té- n- de. O cle- mens: O

pi- a: O dul- cis Vir- go Ma- rí- a.

Support for ancient notation includes features for mensural notation, Gregorian chant notation, and Kievan square notation. These features can be accessed either by modifying style properties of graphical objects such as note heads and rests, or by using one of the pre-defined contexts for these styles.

Many graphical objects, such as note heads and flags, accidentals, time signatures, and rests, provide a `style` property, which can be changed to emulate several different styles of ancient notation. See

- [Mensural note heads], pagina 438,
- [Mensural accidentals and key signatures], pagina 440,
- [Mensural rests], pagina 439,
- [Mensural clefs], pagina 436,
- [Gregorian clefs], pagina 443,
- [Mensural flags], pagina 439,
- [Mensural time signatures], pagina 437.

Some notational concepts are introduced specifically for ancient notation,

- [Custodes], pagina 434,
- [Divisiones], pagina 444,
- [Ligatures], pagina 434.

## Vedi anche

Music Glossary: Sezione “custos” in *Glossario Musicale*, Sezione “ligature” in *Glossario Musicale*, Sezione “mensural notation” in *Glossario Musicale*.

Notation Reference: [Mensural note heads], pagina 438, [Mensural accidentals and key signatures], pagina 440, [Mensural rests], pagina 439, [Gregorian clefs], pagina 443, [Mensural flags], pagina 439, [Mensural time signatures], pagina 437, [Custodes], pagina 434, [Divisiones], pagina 444, [Ligatures], pagina 434.

### 2.9.1 Overview of the supported styles

Three styles are available for typesetting Gregorian chant:

- *Editio Vaticana* is a complete style for Gregorian chant, following the appearance of the Solesmes editions, the official chant books of the Vatican since 1904. LilyPond has support for all the notational signs used in this style, including ligatures, *custodes*, and special signs such as the quilisma and the oriscus.
- The *Editio Medicaea* style offers certain features used in the Medicaea (or Ratisbona) editions which were used prior to the Solesmes editions. The most significant differences from the *Vaticana* style are the clefs, which have downward-slanted strokes, and the note heads, which are square and regular.
- The *Hufnagel* (“horseshoe nail”) or *Gothic* style mimics the writing style in chant manuscripts from Germany and Central Europe during the middle ages. It is named after the basic note shape (the *virga*), which looks like a small nail.

Three styles emulate the appearance of late-medieval and renaissance manuscripts and prints of mensural music:

- The *Mensural* style most closely resembles the writing style used in late-medieval and early renaissance manuscripts, with its small and narrow, diamond-shaped note heads and its rests which approach a hand-drawn style.
- The *Neomensural* style is a modernized and stylized version of the former: the note heads are broader and the rests are made up of straight lines. This style is particularly suited, e.g., for incipits of transcribed pieces of mensural music.
- The *Petrucchi* style is named after Ottaviano Petrucci (1466-1539), the first printer to use movable type for music (in his *Harmonice musices odhecaton*, 1501). The style uses larger note heads than the other mensural styles.

*Baroque* and *Classical* are not complete styles but differ from the default style only in some details: certain note heads (*Baroque*) and the quarter rest (*Classical*).

Only the mensural style has alternatives for all aspects of the notation. Thus, there are no rests or flags in the Gregorian styles, since these signs are not used in plainchant notation, and the Petrucci style has no flags or accidentals of its own.

Each element of the notation can be changed independently of the others, so that one can use mensural flags, petrucci note heads, classical rests and vaticana clefs in the same piece, if one wishes.

### Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*, Sezione “flag” in *Glossario Musicale*.

### 2.9.2 Ancient notation—common features

#### Pre-defined contexts

For Gregorian chant and mensural notation, there are pre-defined voice and staff contexts available, which set all the various notation signs to values suitable for these styles. If one is satisfied with these defaults, one can proceed directly with note entry without worrying about the details on how to customize a context. See one of the pre-defined contexts `VaticanaVoice`, `VaticanaStaff`, `MensuralVoice`, and `MensuralStaff`. See further

- [Gregorian chant contexts], pagina 442,
- [Mensural contexts], pagina 435.

## Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*.

Notation Reference: [Gregorian chant contexts], pagina 442, [Mensural contexts], pagina 435.

## Ligatures

A ligature is a graphical symbol that represents at least two distinct notes. Ligatures originally appeared in the manuscripts of Gregorian chant notation to denote ascending or descending sequences of notes on the same syllable. They are also used in mensural notation.

Ligatures are entered by *enclosing* them in `\[` and `\]`. Some ligature styles may need additional input syntax specific for this particular type of ligature. By default, the `LigatureBracket` engraver just puts a square bracket above the ligature.

```
\relative {
  \[ g' c, a' f d' \]
  a g f
  \[ e f a g \]
}
```



Two other ligature styles are available: the *Vaticana* for Gregorian chant, and the *Mensural* for mensural music (only white mensural ligatures are supported for mensural music, and with certain limitations). To use any of these styles, the default `Ligature_bracket_engraver` has to be replaced with one of the specialized ligature engravers in the `Voice` context, as explained in [White mensural ligatures], pagina 441, and [Gregorian square neume ligatures], pagina 446.

## Vedi anche

Music Glossary: Sezione “ligature” in *Glossario Musicale*.

Notation Reference: [White mensural ligatures], pagina 441, [Gregorian square neume ligatures], pagina 446.

## Problemi noti e avvertimenti

Ligatures need special spacing that has not yet been implemented. As a result, there is too much space between ligatures most of the time, and line breaking often is unsatisfactory. Also, lyrics do not correctly align with ligatures.

Accidentals must not be printed within a ligature, but instead need to be collected and printed in front of it.

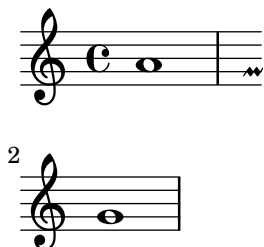
The syntax still uses the deprecated infix style `\[ music expr \]`. For consistency reasons, it will eventually be changed to postfix style `note\[ ... note\]`.

## Custodes

A *custos* (plural: *custodes*; Latin word for “guard”) is a symbol that appears at the end of a staff. It anticipates the pitch of the first note of the following line, thus helping the performer to manage line breaks during performance.

Custodes were frequently used in music notation until the seventeenth century. Nowadays, they have survived only in a few particular forms of musical notation such as contemporary editions of Gregorian chant like the *Editio Vaticana*. There are different custos glyphs used in different flavors of notational style.

For typesetting custodes, just put a `Custos_engraver` into the `Staff` context when declaring the `\layout` block, and change the style of the custos with an `\override` if desired, as shown in the following example:



The custos glyph is selected by the `style` property. The styles supported are `vaticana`, `medicaea`, `hufnagel`, and `mensural`. They are demonstrated in the following fragment.

<code>vaticana</code>	<code>medicaea</code>	<code>hufnagel</code>	<code>mensural</code>
		✓	✓

## Vedi anche

Music Glossary: Sezione “custos” in *Glossario Musicale*.

Snippets: Sezione “Ancient notation” in *Frammenti di codice*.

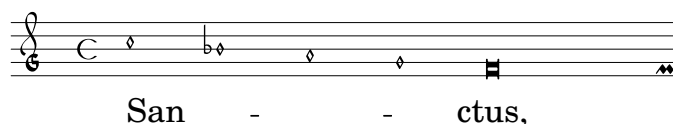
Internals Reference: Sezione “Custos” in *Guida al Funzionamento Interno*.

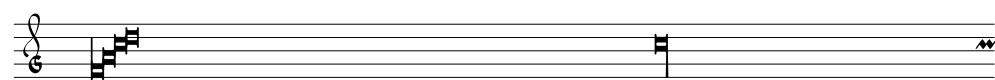
## 2.9.3 Typesetting mensural music

### Mensural contexts

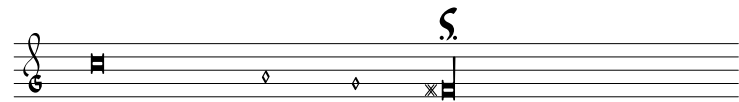
The predefined `MensuralVoice` and `MensuralStaff` contexts can be used to engrave a piece in mensural style. These contexts initialize all relevant context properties and grob properties to proper values, so you can immediately go ahead entering the chant, as the following excerpt demonstrates:

```
\score {
  <<
    \new MensuralVoice = "discantus" \relative {
      \hide Score.BarNumber {
        c''1\melisma bes a g\melismaEnd
        f\breve
        \[ f1\melisma a c\breve d\melismaEnd \]
        c\longa
        c\breve\melisma a1 g1\melismaEnd
        fis\longa^\signumcongruentiae
      }
    }
    \new Lyrics \lyricsto "discantus" {
      San -- ctus, San -- ctus, San -- ctus
    }
  >>
}
```





San - - - ctus,



San - - ctus


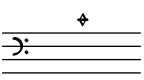
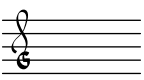
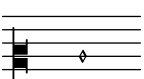


Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*.

Mensural clefs

The following table shows all mensural clefs that are supported via the `\clef` command. Some of the clefs use the same glyph, but differ only with respect to the line they are printed on. In such cases, a trailing number in the name is used to enumerate these clefs, numbered from the lowest to the highest line. You can manually force a clef glyph to be typeset on an arbitrary line, as described in `\Clef`, pagina `\Clef`. The note printed to the right side of each clef in the example column denotes the `c'` with respect to that clef.

Petrucchi used C clefs with differently balanced left-side vertical beams, depending on which staff line it is printed.

Description	Supported Clefs	Example
mensural C clef	<code>mensural-c1</code> , <code>mensural-c2</code> , <code>mensural-c3</code> , <code>mensural-c4</code> , <code>mensural-c5</code>	
mensural F clef	<code>mensural-f</code>	
mensural G clef	<code>mensural-g</code>	
black mensural C clef	<code>blackmensural-c1</code> , <code>blackmensural-c2</code> , <code>blackmensural-c3</code> , <code>blackmensural-c4</code> , <code>blackmensural-c5</code>	
neomensural C clef	<code>neomensural-c1</code> , <code>neomensural-c2</code> , <code>neomensural-c3</code> , <code>neomensural-c4</code>	
petrucci style C clefs, for use on different staff lines (the example shows the 2nd staff line C clef)	<code>petrucci-c1</code> , <code>petrucci-c2</code> , <code>petrucci-c3</code> , <code>petrucci-c4</code> , <code>petrucci-c5</code>	

petrucci style F clefs, for use on different staff lines (the example shows the 3rd staff line F clef)

`petrucci-f3`, `petrucci-f4`,  
`petrucci-f5`



petrucci style G clef

`petrucci-g`



## Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*, Sezione “clef” in *Glossario Musicale*.


Notation Reference: [\[Clef\]](#), pagina [\[Clef\]](#).


## Problemi noti e avvertimenti


The mensural g clef is mapped to the Petrucci g clef.

## Mensural time signatures

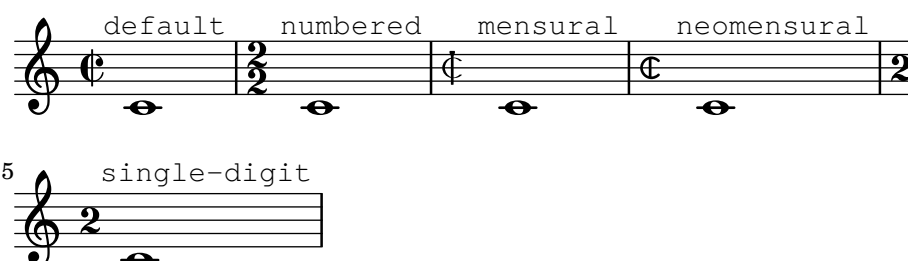
There is limited support for mensuration signs (which are similar to, but not exactly the same as time signatures). The glyphs are hard-wired to particular time fractions. In other words, to get a particular mensuration sign with the `\time n/m` command, `n` and `m` have to be chosen according to the following table

`\time 4/4` `\time 2/2` `\time 6/4` `\time 6/8`  


`\time 3/2` `\time 3/4` `\time 9/4` `\time 9/8`  


`\time 4/8` `\time 2/4`  


Use the `style` property of grob `TimeSignature` to select ancient time signatures. Supported styles are `neomensural` and `mensural`. The above table uses the `neomensural` style. The following examples show the differences in style:



[\[Time signature\]](#), pagina [\[Time signature\]](#), gives a general introduction to the use of time signatures.

## Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*.

Notation Reference: [\[Time signature\]](#), pagina [\[Time signature\]](#).

## Problemi noti e avvertimenti

Ratios of note durations cannot change with the time signature, as those are not constant. For example, the ratio of 1 breve = 3 semibreves (*tempus perfectum*) can be made by hand, by setting

```
breveTP = #(ly:make-duration -1 0 3/2)
...
{ c\breveTP f1 }
```

This sets `breveTP` to  $3/2$  times 2 = 3 times a whole note.

The `mensural68alt` and `neomensural68alt` symbols (alternate symbols for 6/8) are not addressable with `\time`. Use `\markup {\musicglyph #"timesig.mensural68alt" }` instead.

## Mensural note heads

For ancient notation, a note head style other than the `default` style may be chosen. This is accomplished by setting the `style` property of the `NoteHead` object to `baroque`, `neomensural`, `mensural`, `petrucci`, `blackpetrucci` or `semipetrucci`.

The `baroque` style differs from the `default` style by:

- Providing a `maxima` note head, and
- Using a square shape for `\breve` note heads.

The `neomensural`, `mensural`, and `petrucci` styles differ from the `baroque` style by:

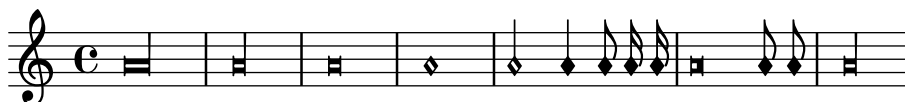
- Using rhomboidal heads for semibreves and all smaller durations, and
- Centering the stems on the note heads.

The `blackpetrucci` style produces note heads usable in black mensural notation or coloratio sections in white mensural notation. Because note head style does not influence flag count, in this style a semiminima should be notated as `a8*2`, not `a4`, otherwise it will look like a minima. The multiplier can be different if coloratio is used e.g. to notate triplets.

Use `semipetrucci` style to draw half-colored note heads (breves, longas and maximas).

The following example demonstrates the `petrucci` style:

```
\set Score.skipBars = ##t
\autoBeamOff
\override NoteHead.style = #'petrucci
a'\maxima a'\longa a'\breve a'1 a'2 a'4 a'8 a'16 a'
\override NoteHead.style = #'semipetrucci
a'\breve*5/6
\override NoteHead.style = #'blackpetrucci
a'8*4/3 a'
\override NoteHead.style = #'petrucci
a'\longa
```



[\[Note head styles\]](#), pagina [\[Note head styles\]](#), gives an overview of all available note head styles.

## Vedi anche

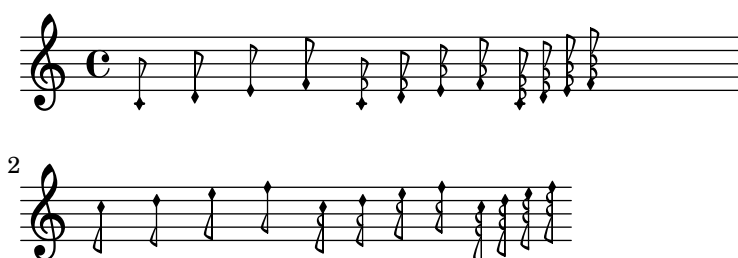
Music Glossary: Sezione “mensural notation” in *Glossario Musicale*, Sezione “note head” in *Glossario Musicale*.

Notation Reference: [\[Note head styles\]](#), pagina [\[Note head styles\]](#).

## Mensural flags

Use the `flag-style` property of grob `Stem` to select ancient flags. Besides the `default` flag style, only the `mensural` style is supported.

```
\relative c' {
  \override Flag.style = #'mensural
  \override Stem.thickness = #1.0
  \override NoteHead.style = #'mensural
  \autoBeamOff
  c8 d e f c16 d e f c32 d e f s8
  c'8 d e f c16 d e f c32 d e f
}
```



Note that the innermost flare of each mensural flag is vertically aligned with a staff line.

There is no particular flag style for neo-mensural or Petrucci notation. There are no flags in Gregorian chant notation.

## Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*, Sezione “flag” in *Glossario Musicale*.

## Problemi noti e avvertimenti

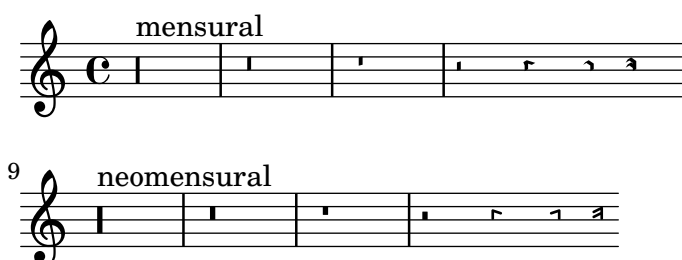
Vertically aligning each flag with a staff line assumes that stems always end either exactly on or exactly in the middle of two staff lines. This may not always be true when using advanced layout features of classical notation (which however are typically out of scope for mensural notation).

## Mensural rests

Use the `style` property of grob `Rest` to select ancient rests. Supported ancient styles are `neomensural`, and `mensural`.

The following example demonstrates these styles:

```
\set Score.skipBars = ##t
\override Rest.style = #'mensural
r\longa^"mensural" r\breve r1 r2 r4 r8 r16 s \break
\override Rest.style = #'neomensural
r\longa^"neomensural" r\breve r1 r2 r4 r8 r16
```





There are no 32nd and 64th rests specifically for the mensural or neo-mensural styles. Rests from the default style are used.

## Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*.

Notation Reference: [\[Rests\]](#), pagina [\[undefined\]](#).

Snippets: Sezione “Ancient notation” in *Frammenti di codice*.

## Problemi noti e avvertimenti

The glyph for the maxima rest in mensural style is actually a perfect longa rest; use two (or three) longa rests to print a maxima rest. Longa rests are not grouped automatically, so have to be done manually by using pitched rests.

## Mensural accidentals and key signatures

The `mensural` style provides a sharp and a flat sign different from the default style. Mensural notation rarely used a natural sign: instead the appropriate sharp or flat is used. For example, a B natural in the key of F major would be indicated with a sharp. However, if specifically called for, the natural sign is taken from the `vaticana` style.

### mensural

♭ ✖

The style for accidentals and key signatures is controlled by the `glyph-name-alist` property of the grobs `Accidental` and `KeySignature`, respectively; e.g.:

```
\override Staff.Accidental.glyph-name-alist =
  #alteration-mensural-glyph-name-alist
```

## Vedi anche

Music Glossary: Sezione “mensural notation” in *Glossario Musicale*, Sezione “Pitch names” in *Glossario Musicale*, Sezione “accidental” in *Glossario Musicale*, Sezione “key signature” in *Glossario Musicale*.

Notation Reference: [\[Pitches\]](#), pagina [\[undefined\]](#), [\[Accidentals\]](#), pagina [\[undefined\]](#), [\[Automatic accidentals\]](#), pagina [\[undefined\]](#), [\[Key signature\]](#), pagina [\[undefined\]](#).

Internals Reference: Sezione “KeySignature” in *Guida al Funzionamento Interno*.

## Annotational accidentals (*musica ficta*)

In European music from before about 1600, singers were expected to chromatically alter notes at their own initiative according to certain rules. This is called *musica ficta*. In modern transcriptions, these accidentals are usually printed over the note.

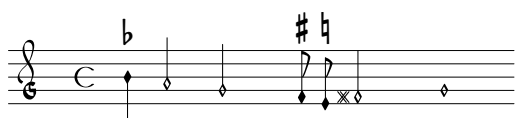
Support for such suggested accidentals is included, and can be switched on by setting `suggestAccidentals` to true.

```
\relative {
  fis' gis
  \set suggestAccidentals = ##t
  ais bis
}
```



This will treat *every* subsequent accidental as *musica ficta* until it is unset with `\set suggestAccidentals = ##f`. A more practical way is to use `\once \set suggestAccidentals = ##t`, which can even be defined as a convenient shorthand:

```
ficta = { \once \set suggestAccidentals = ##t }
\score { \relative
  \new MensuralVoice {
    \once \set suggestAccidentals = ##t
    bes'4 a2 g2 \ficta fis8 \ficta e! fis2 g1
  }
}
```



## Vedi anche

Internals Reference: Sezione “Accidental\_engraver” in *Guida al Funzionamento Interno*, Sezione “AccidentalSuggestion” in *Guida al Funzionamento Interno*.

## White mensural ligatures

There is limited support for white mensural ligatures.

To engrave white mensural ligatures, in the layout block, replace the `Ligature_bracket_engraver` with the `Mensural_ligature_engraver` in the `Voice` context:

```
\layout {
  \context {
    \Voice
    \remove "Ligature_bracket_engraver"
    \consists "Mensural_ligature_engraver"
  }
}
```

There is no additional input language to describe the shape of a white mensural ligature. The shape is rather determined solely from the pitch and duration of the enclosed notes. While this approach may take a new user a while to get accustomed to, it has the great advantage that the full musical information of the ligature is known internally. This is not only required for correct MIDI output, but also allows for automatic transcription of the ligatures.

At certain places two consecutive notes can be represented either as two squares or as an oblique parallelogram (flexa shape). In such cases the default is the two squares, but a flexa can be required by setting the `ligature-flexa` property of the *second* note head. The length of a flexa can be set by the note head property `flexa-width`.

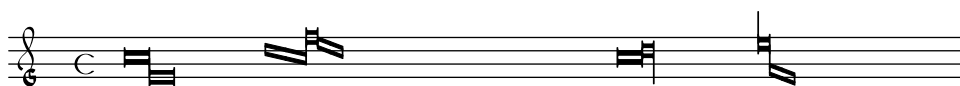
For example,

```
\score {
  \relative {
    \set Score.timing = ##f
    \set Score.defaultBarType = "-"
    \override NoteHead.style = #'petrucci
    \override Staff.TimeSignature.style = #'mensural
```

```

\clef "petrucci-g"
\[ c''\maxima g \]
\[ d'\longa
  \override NoteHead.ligature-flexa = ##t
  \once \override NoteHead.flexa-width = #3.2
  c\breve f e d \]
\[ c\maxima d\longa \]
\[ e1 a, g\breve \]
}
\layout {
  \context {
    \Voice
    \remove "Ligature_bracket_engraver"
    \consists "Mensural_ligature_engraver"
  }
}
}

```



Without replacing `Ligature_bracket_engraver` with `Mensural_ligature_engraver`, the same music looks as follows:



## Vedi anche

Music Glossary: Sezione “ligature” in *Glossario Musicale*.

Notation Reference: [Gregorian square neume ligatures], pagina 446, [Ligatures], pagina 434.

## Problemi noti e avvertimenti

Horizontal spacing of ligatures may be poor. Accidentals may collide with previous notes.

### 2.9.4 Typesetting Gregorian chant

When typesetting a piece in Gregorian chant notation, the `Vaticana_ligature_engraver` automatically selects the proper note heads, so there is no need to explicitly set the note head style. Still, the note head style can be set, e.g., to `vaticana_punctum` to produce punctum neumes. Similarly, the `Mensural_ligature_engraver` automatically assembles mensural ligatures.

## Vedi anche

Music Glossary: Sezione “ligature” in *Glossario Musicale*.

Notation Reference: [White mensural ligatures], pagina 441, [Ligatures], pagina 434.

## Gregorian chant contexts

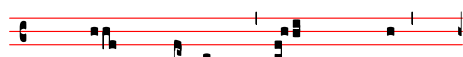
The predefined `VaticanaVoice` and `VaticanaStaff` can be used to engrave a piece of Gregorian chant in the style of the Editio Vaticana. These contexts initialize all relevant context properties and grob properties to proper values, so you can immediately go ahead entering the chant, as the following excerpt demonstrates:

```
\include "gregorian.ly"
```

```

\score {
  <<
    \new VaticanaVoice = "cantus" {
      \[ c'\melisma c' \flexa a \]
      \[ a \flexa \deminutum g\melismaEnd \]
      f \divisioMinima
      \[ f\melisma \pes a c' c' \pes d'\melismaEnd \]
      c' \divisioMinima \break
      \[ c'\melisma c' \flexa a \]
      \[ a \flexa \deminutum g\melismaEnd \] f \divisioMinima
    }
    \new Lyrics \lyricsto "cantus" {
      San- ctus, San- ctus, San- ctus
    }
  >>
}

```




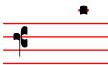

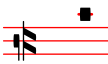
San- ctus, San- ctus,



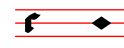
San- ctus

## Gregorian clefs

The following table shows all Gregorian clefs that are supported via the `\clef` command. Some of the clefs use the same glyph, but differ only with respect to the line they are printed on. In such cases, a trailing number in the name is used to enumerate these clefs, numbered from the lowest to the highest line. Still, you can manually force a clef glyph to be typeset on an arbitrary line, as described in [\[Clef\]](#), pagina [\(undefined\)](#). The note printed to the right side of each clef in the example column denotes the `c'` with respect to that clef.

Description	Supported Clefs	Example
Editio Vaticana style do clef	<code>vaticana-do1</code> , <code>vaticana-do2</code> , <code>vaticana-do3</code>	
Editio Vaticana style fa clef	<code>vaticana-fa1</code> , <code>vaticana-fa2</code>	
Editio Medicaea style do clef	<code>medicaea-do1</code> , <code>medicaea-do2</code> , <code>medicaea-do3</code>	
Editio Medicaea style fa clef	<code>medicaea-fa1</code> , <code>medicaea-fa2</code>	

hufnagel style do clef

hufnagel-do1, hufnagel-do2,  
hufnagel-do3

hufnagel style fa clef

hufnagel-fa1, hufnagel-fa2



hufnagel style combined do/fa clef

hufnagel-do-fa



## Vedi anche

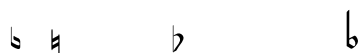
Music Glossary: Sezione “clef” in *Glossario Musicale*.

Notation Reference: [\[Clef\]](#), pagina [\[Clef\]](#).

## Gregorian accidentals and key signatures

Accidentals for the three different Gregorian styles are available:

### vaticana medicaea hufnagel



As shown, not all accidentals are supported by each style. When trying to access an unsupported accidental, LilyPond will switch to a different style.

The style for accidentals and key signatures is controlled by the `glyph-name-alist` property of the grobs `Accidental` and `KeySignature`, respectively; e.g.:

```
\override Staff.Accidental.glyph-name-alist =
  #alteration-mensural-glyph-name-alist
```

## Vedi anche

Music Glossary: Sezione “accidental” in *Glossario Musicale*, Sezione “key signature” in *Glossario Musicale*.

Notation Reference: [\[Pitches\]](#), pagina [\[Pitches\]](#), [\[Accidentals\]](#), pagina [\[Accidentals\]](#), [\[Automatic accidentals\]](#), pagina [\[Automatic accidentals\]](#), [\[Key signature\]](#), pagina [\[Key signature\]](#).

Internals Reference: Sezione “KeySignature” in *Guida al Funzionamento Interno*.

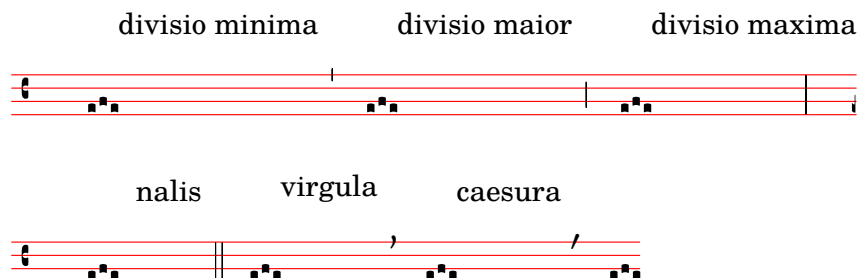
## Divisiones

There are no rests in Gregorian chant notation; instead, it uses [\[Divisiones\]](#), pagina 444.

A *divisio* (plural: *divisiones*; Latin word for ‘division’) is a staff context symbol that is used to indicate the phrase and section structure of Gregorian music. The musical meaning of *divisio minima*, *divisio maior*, and *divisio maxima* can be characterized as short, medium, and long pause, somewhat like the breath marks from [\[Breath marks\]](#), pagina [\[Breath marks\]](#). The *finalis* sign not only marks the end of a chant, but is also frequently used within a single antiphonal/responsorial chant to mark the end of each section.

To use divisiones, include the file `gregorian.ly`. It contains definitions that you can apply by just inserting `\divisioMinima`, `\divisioMaior`, `\divisioMaxima`, and `\finalis` at proper

places in the input. Some editions use *virgula* or *caesura* instead of *divisio minima*. Therefore, `gregorian.ly` also defines `\virgula` and `\caesura`



## Comandi predefiniti

`\virgula`, `\caesura`, `\divisioMinima`, `\divisioMaior`, `\divisioMaxima`, `\finalis`.

## Vedi anche

Music Glossary: Sezione “caesura” in *Glossario Musicale*, Sezione “divisio” in *Glossario Musicale*.

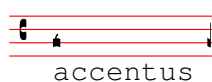
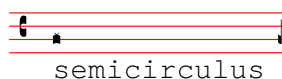
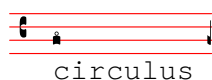
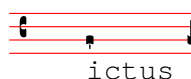
Notation Reference: `\Breath marks`, pagina `\Breath marks`.

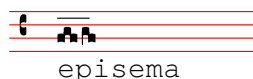
Installed Files: `ly/gregorian.ly`.

## Gregorian articulation signs

In addition to the standard articulation signs described in section `\Articulations and ornamentations`, articulation signs specifically designed for use with notation in *Editio Vaticana* style are provided.

```
\include "gregorian.ly"
\score {
  \new VaticanaVoice {
    \override TextScript.font-family = #'typewriter
    \override TextScript.font-shape = #'upright
    \override Script.padding = #-0.1
    a\ictus_"ictus " \bar "" \break
    a\circulus_"circulus " \bar "" \break
    a\semicirculus_"semicirculus " \bar "" \break
    a\accentus_"accentus " \bar "" \break
    \[ a_"episema" \epistemInitium \pes b \flexa a b \epistemFinis \flexa a \]
  }
}
```





## Vedi anche

Notation Reference: [\[Articulations and ornamentations\]](#), pagina [\[undefined\]](#).

Snippets: Sezione “Ancient notation” in *Frammenti di codice*.

Internals Reference: Sezione “Episema” in *Guida al Funzionamento Interno*, Sezione “EpisemaEvent” in *Guida al Funzionamento Interno*, Sezione “Episema\_engraver” in *Guida al Funzionamento Interno*, Sezione “Script” in *Guida al Funzionamento Interno*, Sezione “ScriptEvent” in *Guida al Funzionamento Interno*, Sezione “Script\_engraver” in *Guida al Funzionamento Interno*.

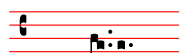
## Problemi noti e avvertimenti

Some articulations are vertically placed too closely to the corresponding note heads.

### Augmentum dots (*morae*)

Augmentum dots, also called *morae*, are added with the music function `\augmentum`. Note that `\augmentum` is implemented as a unary music function rather than as head prefix. It applies to the immediately following music expression only. That is, `\augmentum \virga c` will have no visible effect. Instead, say `\virga \augmentum c` or `\augmentum {\virga c}`. Also note that you can say `\augmentum {a g}` as a shortcut for `\augmentum a \augmentum g`.

```
\include "gregorian.ly"
\score {
  \new VaticanaVoice {
    \[ \augmentum a \flexa \augmentum g \]
    \augmentum g
  }
}
```



## Vedi anche

Notation Reference: [\[Breath marks\]](#), pagina [\[undefined\]](#).

Internals Reference: Sezione “BreathingSign” in *Guida al Funzionamento Interno*.

Snippets: Sezione “Ancient notation” in *Frammenti di codice*.

## Gregorian square neume ligatures

There is limited support for Gregorian square neumes notation (following the style of the Editio Vaticana). Core ligatures can already be typeset, but essential issues for serious typesetting are still lacking, such as (among others) horizontal alignment of multiple ligatures, lyrics alignment, and proper handling of accidentals.

The support for Gregorian neumes is enabled by `\includeing gregorian.ly` at the beginning of the file. This makes available a number of extra commands to produce the neume symbols used in plainchant notation.

Note heads can be *modified* and/or *joined*.

- The shape of the note head can be modified by *prefixing* the note name with any of the following commands: `\virga`, `\strophæ`, `\inclinatum`, `\auctum`, `\descendens`, `\ascendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`.
- Ligatures, properly speaking (i.e. notes joined together), are produced by placing one of the joining commands `\pes` or `\flexa`, for upwards and downwards movement, respectively, *between* the notes to be joined.

A note name without any qualifiers will produce a *punctum*. All other neumes, including the single-note neumes with a different shape such as the *virga*, are in principle considered as ligatures and should therefore be placed between `\[...]`.

Single-note neumes:

- The *punctum* is the basic note shape (in the *Vaticana* style: a square with some curvature for typographical finesse). In addition to the regular *punctum*, there is also the oblique *punctum inclinatum*, produced with the prefix `\inclinatum`. The regular *punctum* can be modified with `\cavum`, which produces a hollow note, and `\linea`, which draws vertical lines on either side of the note.
- The *virga* has a descending stem on the right side. It is produced by the modifier `\virga`.

Ligatures

Unlike most other neumes notation systems, the typographical appearance of ligatures is not directly dictated by the input commands, but follows certain conventions dependent on musical meaning. For example, a three-note ligature with the musical shape low-high-low, such as `\[ a \pes b \flexa g ]`, produces a Torculus consisting of three Punctum heads, while the shape high-low-high, such as `\[ a \flexa g \pes b ]`, produces a Porrectus with a curved flexa shape and only a single Punctum head. There is no command to explicitly typeset the curved flexa shape; the decision of when to typeset a curved flexa shape is based on the musical input. The idea of this approach is to separate the musical aspects of the input from the notation style of the output. This way, the same input can be reused to typeset the same music in a different style of Gregorian chant notation.

Liquescent neumes

Another main category of notes in Gregorian chant is the so-called liquescent neumes. They are used under certain circumstances at the end of a syllable which ends in a ‘liquescent’ letter, i.e. the sounding consonants that can hold a tone (the nasals, l, r, v, j, and their diphthong equivalents). Thus, the liquescent neumes are never used alone (although some of them can be produced), and they always fall at the end of a ligature.

Liquescent neumes are represented graphically in two different, more or less interchangeable ways: with a smaller note or by ‘twisting’ the main note upwards or downwards. The first is produced by making a regular `pes` or `flexa` and modifying the shape of the second note: `\[ a \pes \deminutum b ]`, the second by modifying the shape of a single-note neume with `\auctum` and one of the direction markers `\descendens` or `\ascendens`, e.g., `\[ \auctum \descendens a ]`.

Special signs

A third category of signs is made up of a small number of signs with a special meaning (which, incidentally, in most cases is only vaguely known): the *quilisma*, the *oriscus*, and the *strophicus*. These are all produced by prefixing a note name with the corresponding modifier, `\quilisma`, `\oriscus`, or `\strophica`.

Virtually, within the ligature delimiters `\[` and `\]`, any number of heads may be accumulated to form a single ligature, and head prefixes like `\pes`, `\flexa`, `\virga`, `\inclinatum`, etc. may be mixed in as desired. The use of the set of rules that underlies the construction of the ligatures in the above table is accordingly extrapolated. This way, infinitely many different ligatures can be created.

Note that the use of these signs in the music itself follows certain rules, which are not checked by LilyPond. E.g., the *quilisma* is always the middle note of an ascending ligature, and usually falls on a half-tone step, but it is perfectly possible, although incorrect, to make a single-note quilisma.








In addition to the note signs, `gregorian.ly` also defines the commands `\versus`, `\responsum`, `\ij`, `\iij`, `\IJ`, and `\IIJ`, that will produce the corresponding characters, e.g., for use in lyrics,



as section markers, etc. These commands use special Unicode characters and will only work if a font is used which supports them.

The following table shows a limited, but still representative pool of Gregorian ligatures, together with the code fragments that produce the ligatures. The table is based on the extended neumes table of the 2nd volume of the *Antiphonale Romanum* (*Liber Hymnarius*), published 1983 by the monks of Solesmes. The first column gives the name of the ligature, with the main form in boldface and the liquescent forms in italics. The third column shows the code fragment that produces this ligature, using **g**, **a**, and **b** as example pitches.

### Single-note neums

Basic and <i>Liquescent</i> forms	Output	LilyPond code
<b>Punctum</b>		<code>\[ b \]</code>
		<code>\[ \cavum b \]</code>
		<code>\[ \linea b \]</code>
<i>Punctum Auctum Ascendens</i>		<code>\[ \auctum \ascendens b \]</code>
<i>Punctum Auctum Descendens</i>		<code>\[ \auctum \descendens b \]</code>
<b>Punctum inclinatum</b>		<code>\[ \inclinatum b \]</code>
<i>Punctum Inclinatum Auctum</i>		<code>\[ \inclinatum \auctum b \]</code>

*Punctum Inclinatum Parvum*

\[ \inclinatum \deminutum b \]

•

**Virga**

┐

**Two-note ligatures****Clivis vel Flexa**

\[ b \flexa g \]

┌┐

*Clivis Aucta Descendens*\[ b \flexa \auctum \descendens  
g \]

┌┐

*Clivis Aucta Ascendens*\[ b \flexa \auctum \ascendens  
g \]

┐┌

*Cephalicus*

\[ b \flexa \deminutum g \]

┌┐

**Podatus/Pes**

\[ g \pes b \]

┐┐

*Pes Auctus Descendens*\[ g \pes \auctum \descendens b  
\]

┐┐

*Pes Auctus Ascendens*\[ g \pes \auctum \ascendens b  
\]

┐┐

*Epiphonus*

\[ g \pes \deminutum b \]

*Pes Initio Debilis*

\[ \deminutum g \pes b \]

*Pes Auctus Descendens Initio Debilis*\[ \deminutum g \pes \auctum  
\descendens b \]**Multi-note ligatures****Torculus**

\[ a \pes b \flexa g \]

*Torculus Auctus Descendens*\[ a \pes b \flexa \auctum  
\descendens g \]*Torculus Deminutus*\[ a \pes b \flexa \deminutum g  
\]*Torculus Initio Debilis*\[ \deminutum a \pes b \flexa g  
\]*Torculus Auctus Descendens Initio  
Debilis*\[ \deminutum a \pes b \flexa  
\auctum \descendens g \]*Torculus Deminutus Initio Debilis*\[ \deminutum a \pes b \flexa  
\deminutum g \]

**Porrectus**

\[ a \flexa g \pes b \]

*Porrectus Auctus Descendens*\[ a \flexa g \pes \auctum  
\descendens b \]*Porrectus Deminutus*\[ a \flexa g \pes \deminutum b  
\]**Climacus**\[ \virga b \inclinatum a  
\inclinatum g \]*Climacus Auctus*\[ \virga b \inclinatum a  
\inclinatum \auctum g \]*Climacus Deminutus*\[ \virga b \inclinatum a  
\inclinatum \deminutum g \]**Scandicus**

\[ g \pes a \virga b \]

*Scandicus Auctus Descendens*\[ g \pes a \pes \auctum  
\descendens b \]*Scandicus Deminutus*

\[ g \pes a \pes \deminutum b \]

**Special Signs**

**Quilisma**

\[ g \pes \quilisma a \pes b \]

*Quilisma Pes Auctus Descendens*\[ \quilisma g \pes \auctum  
\descendens b \]**Oriscus**

\[ \oriscus b \]

*Pes Quassus*

\[ \oriscus g \pes \virga b \]

*Pes Quassus Auctus Descendens*\[ \oriscus g \pes \auctum  
\descendens b \]**Salicus**

\[ g \oriscus a \pes \virga b \]

*Salicus Auctus Descendens*\[ g \oriscus a \pes \auctum  
\descendens b \]**(Apo)stropha**

\[ \stropha b \]

*Stropha Aucta*

\[ \stropha \auctum b \]

**Bistropha**

\[ \stropha b \stropha b \]



**Tristropha**


```
\[ \stropha b \stropha b
\stropha b \]
```

*Trigonus*


```
\[ \stropha b \stropha b
\stropha a \]
```

**Comandi predefiniti**

The following head prefixes are supported: `\virga`, `\stropha`, `\inclinatum`, `\auctum`, `\descendens`, `\ascendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`.

Head prefixes can be accumulated, though restrictions apply. For example, either `\descendens` or `\ascendens` can be applied to a head, but not both to the same head.

Two adjacent heads can be tied together with the `\pes` and `\flexa` infix commands for a rising and falling line of melody, respectively.

Use the unary music function `\augmentum` to add augmentum dots.

**Vedi anche**

Music Glossary: Sezione “ligature” in *Glossario Musicale*.

Notation Reference: [Gregorian square neume ligatures], pagina 446, [White mensural ligatures], pagina 441, [Ligatures], pagina 434.

**Problemi noti e avvertimenti**

When an `\augmentum` dot appears at the end of the last staff within a ligature, it is sometimes vertically placed wrong. As a workaround, add an additional skip note (e.g., `s8`) as last note of the staff.

`\augmentum` should be implemented as a head prefix rather than a unary music function, such that `\augmentum` can be intermixed with head prefixes in arbitrary order.

**2.9.5 Typesetting Kievan square notation****Kievan contexts**

As with Mensural and Gregorian notation, the predefined `KievanVoice` and `KievanStaff` contexts can be used to engrave a piece in square notation. These contexts initialize all relevant context properties and grob properties to proper values, so you can immediately go ahead entering the chant:

```
% Font settings for Cyrillic
\paper {
  #(define fonts
    (set-global-fonts
      #:roman "Linux Libertine O,serif"
    ))
}

\score {
  <<
  \new KievanVoice = "melody" \relative c' {
```

```

\cadenzaOn
  c4 c c c c2 b\longa
  \bar "k"
}
\new Lyrics \lyricsto "melody" {
  Го -- спо -- ди по -- ми -- луй.
}
>>
}

```



## Vedi anche

Music Glossary: Sezione “kievan notation” in *Glossario Musicale*.

## Problemi noti e avvertimenti

LilyPond supports Kievan notation of the Synodal style, as used in the corpus of chantbooks printed by the Russian Holy Synod in the 1910’s and recently reprinted by the Moscow Patriarchate Publishing House. LilyPond does not support the older (less common) forms of Kievan notation that were used in Galicia to notate Rusyn plainchant.

## Kievan clefs

There is only one clef used in Kievan notation (the Tse-fa-ut Clef). It is used to indicate the position of c:

```

\clef "kievan-do"
\kievanOn
c'

```



## Vedi anche

Music Glossary: Sezione “kievan notation” in *Glossario Musicale*, Sezione “clef” in *Glossario Musicale*.

Notation Reference: [\[Clef\]](#), pagina [\[Clef\]](#).

## Kievan notes

For Kievan square notation, the appropriate note head style needs to be chosen and the flags and stems need to be turned off. This is accomplished by calling the `\kievanOn` function, which sets the appropriate properties of the note head, stems, and flags. Once Kievan note heads are not needed, these properties can be reverted by calling the `\kievanOff` function.

The Kievan final note, which usually comes at the end of a piece of music, may be selected by setting the duration to `\longa`. The Kievan recitative mark, used to indicate the chanting of several syllables on one note, may be selected by setting the duration to `\breve`. The following example demonstrates the various Kievan note heads:

```

\autoBeamOff
\cadenzaOn

```

```
\kievanOn
b'1 b'2 b'4 b'8 b'\breve b'\longa
\kievanOff
b'2
```



## Vedi anche

Music Glossary: Sezione “kievan notation” in *Glossario Musicale*, Sezione “note head” in *Glossario Musicale*.

Notation Reference: [\[Note head styles\]](#), pagina [\[undefined\]](#).

## Problemi noti e avvertimenti

LilyPond automatically determines if the stem up or stem down form of a note is drawn. When setting chant in square notation, however, it is customary to have the stems point in the same direction within a single melisma. This can be done manually by setting the `direction` property of the `Stem` object.

## Kievan accidentals

The `kievan` style for accidentals is selected with the `glyph-name-alist` property of the grob `Accidental`. The `kievan` style provides a sharp and a flat sign different from the default style. There is no natural sign in Kievan notation. The sharp sign is not used in Synodal music but may occur in earlier manuscripts. It has been included primarily for the sake of compatibility.

```
\clef "kievan-do"
\override Accidental.glyph-name-alist =
  #alteration-kievan-glyph-name-alist
bes' dis'
```



## Vedi anche

Music Glossary: Sezione “kievan notation” in *Glossario Musicale*, Sezione “accidental” in *Glossario Musicale*.

Notation Reference: [\[undefined\]](#) [\[Accidentals\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Automatic accidentals\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[The Feta font\]](#), pagina [\[undefined\]](#),

## Kievan bar line

A decorative figure is commonly placed at the end of a piece of Kievan notation, which may be called the Kievan final bar line. It can be invoked as `\bar "k"`.

```
\kievanOn
\clef "kievan-do"
c' \bar "k"
```





## Vedi anche

⟨undefined⟩ [Bars], pagina ⟨undefined⟩, ⟨undefined⟩ [The Feta font], pagina ⟨undefined⟩,

## Kievan melismata

Notes within a Kievan melisma are usually placed close to each other and the melismata separated by whitespace. This is done to allow the chanter to quickly identify the melodic structures of Znamenny chant. In LilyPond, melismata are treated as ligatures and the spacing is implemented by the `Kievan_ligature_engraver`.

When the `KievanVoice` and `KievanStaff` contexts are used, the `Kievan_ligature_engraver` is enabled by default. In other contexts, it can be invoked by replacing the `Ligature_bracket_engraver` with the `Kievan_ligature_engraver` in the layout block:

```
\layout {
  \context {
    \Voice
    \remove "Ligature_bracket_engraver"
    \consists "Kievan_ligature_engraver"
  }
}
```

The spacing between the notes within a Kievan ligature can be controlled by setting the `padding` property of the `KievanLigature`.

The following example demonstrates the use of Kievan ligatures:

```
% Font settings for Cyrillic
\paper {
  #(define fonts
    (set-global-fonts
      #:roman "Linux Libertine O,serif"
    ))
}

\score {
  <<
    \new KievanVoice = "melody" \relative c' {
      \cadenzaOn
      e2 \[ e4( d4 ) \] \[ c4( d e d ) \] e1 \bar "k"
    }
    \new Lyrics \lyricsto "melody" {
      Га -- вpi -- и -- лу
    }
  >>
}
```



## Vedi anche

Music Glossary: Sezione “ligature” in *Glossario Musicale*.

Notation Reference: [White mensural ligatures], pagina 441, [Gregorian square neume ligatures], pagina 446, [Ligatures], pagina 434.

## Problemi noti e avvertimenti

Horizontal spacing of ligatures is poor.

### 2.9.6 Working with ancient music—scenarios and solutions

Working with ancient music frequently involves particular tasks which differ considerably from the modern notation for which LilyPond is designed. In the rest of this section, a number of typical scenarios are outlined, with suggestions of solutions. These involve:

- how to make incipits (i.e. prefatory material to indicate what the original has looked like) to modern transcriptions of mensural music;
- how to achieve the *Mensurstriche* layout frequently used for modern transcriptions of polyphonic music;
- how to transcribe Gregorian chant in modern notation;
- how to generate both ancient and modern notation from the same source.

## Incipits

It is customary when transcribing mensural music into modern notation to place an indication of how the initial rests and note or notes of the original version appeared - including the original clefs. This is called an *incipit*. The `\incipit` command uses the `indent` of the main staff to set the width occupied by the incipit, and `incipit-width` to set the width of the incipit staff.

```
\score {
  \new Staff <<
    \new Voice = Tenor {
      \set Staff.instrumentName = #"Tenor"
      \override Staff.InstrumentName.self-alignment-X = #RIGHT
      \incipit { \clef "mensural-c4" \key f \major r\breve r1 c'1 }
      \clef "treble_8"
      \key f \major
      R1 r2 c'2 |
      a4. c'8
    }
    \new Lyrics \lyricsto Tenor { Cyn -- thia your }
  >>
  \layout
  {
    indent = 5\cm
    incipit-width = 3\cm
  }
}
```



## Problemi noti e avvertimenti

Note that `instrumentName` must be set in the music for the incipit to be produced. If no instrument name is required then use `\set Staff.instrumentName = #""`.

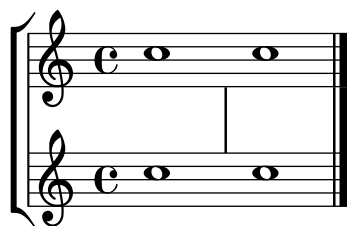
## Mensurstriche layout

*Mensurstriche* ('mensuration lines') is the accepted term for bar lines that are drawn between the staves of a system but not through the staves themselves. It is a common way to preserve the rhythmic appearance of the original, i.e. not having to break syncopated notes at bar lines, while still providing the orientation aids that bar lines give.

La formattazione mensurale, in cui le stanghette non appaiono sui righi ma nello spazio tra i righi, si può ottenere usando `StaffGroup` al posto di `ChoirStaff`. La stanghetta sui righi viene nascosta impostando la proprietà `transparent`.

```
global = {
  \hide Staff.BarLine
  s1 s
  % the final bar line is not interrupted
  \undo \hide Staff.BarLine
  \bar "|."
}

\new StaffGroup \relative c'' {
  <<
    \new Staff { << \global { c1 c } >> }
    \new Staff { << \global { c c } >> }
  >>
}
```



## Transcribing Gregorian chant

Gregorian chant can be transcribed into modern notation with a number of simple tweaks.

**Stems.** Stems can be left out altogether by `\remove`-ing the `Stem_engraver` from the `Voice` context:

```
\layout {
  ...
  \context {
    \Voice
    \remove "Stem_engraver"
  }
}
```

**Timing.** For unmetered chant, there are several alternatives.

The `Time_signature_engraver` can be removed from the `Staff` context without any negative side effects. The alternative, to make it transparent, will leave an empty space in the score, since the invisible signature will still take up space.

In many cases, `\set Score.timing = ##f` will give good results. Another alternative is to use `\cadenzaOn` and `\cadenzaOff`.

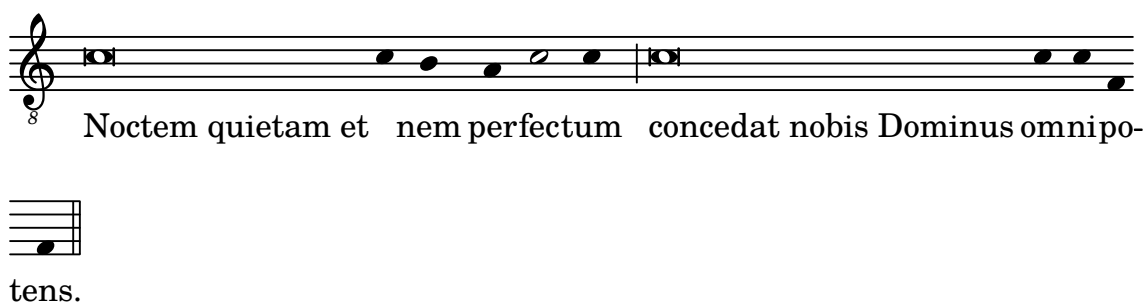
To remove the bar lines, the radical approach is to `\remove` the `Bar_engraver` from the `Staff` context. Again, one may want to use `\hide BarLine` instead, if an occasional barline is wanted.

A common type of transcription is recitativic chant where the repeated notes are indicated with a single breve. The text to the recitation tone can be dealt with in two different ways: either set as a single, left-aligned syllable:

```
\include "gregorian.ly"
chant = \relative {
  \clef "G_8"
  c'\breve c4 b4 a c2 c4 \divisioMaior
  c\breve c4 c f, f \finalis
}

verba = \lyricmode {
  \once \override LyricText.self-alignment-X = #-1
  "Noctem quietam et" fi -- nem per -- fec -- tum
  \once \override LyricText.self-alignment-X = #-1
  "concedat nobis Dominus" om -- ni -- po -- tens.
}

\score {
  \new Staff <<
  \new Voice = "melody" \chant
  \new Lyrics = "one" \lyricsto melody \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Bar_engraver"
    }
    \context {
      \Voice
      \remove "Stem_engraver"
    }
  }
}
```



This works fine, as long as the text doesn't span a line break. If that is the case, an alternative is to add hidden notes to the score, as below.

In some transcription styles, stems are used occasionally, for example to indicate the transition from a single-tone recitative to a fixed melodic gesture. In these cases, one can use either `\hide Stem` or `\override Stem.length = #0` instead of `\remove`-ing the `Stem_engraver` and restore the stem when needed with the corresponding `\undo \hide Stem`.

```
\include "gregorian.ly"
chant = \relative {
```

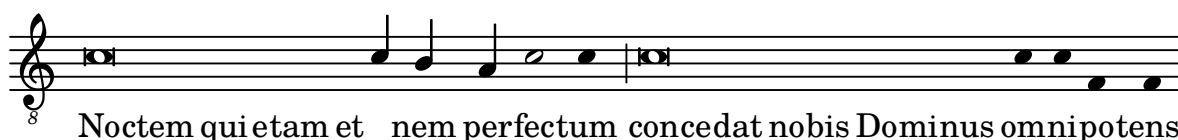
```

\clef "G_8"
\set Score.timing = ##f
\hide Stem
c'\breve \hide NoteHead c c c c c
\undo \hide NoteHead
\undo \hide Stem \stemUp c4 b4 a
\hide Stem c2 c4 \divisioMaior
c\breve \hide NoteHead c c c c c c c
\undo \hide NoteHead c4 c f, f \finalis
}

verba = \lyricmode {
  No -- ctem qui -- e -- tam et fi -- nem per -- fec -- tum
  con -- ce -- dat no -- bis Do -- mi -- nus om -- ni -- po -- tens.
}

\score {
  \new Staff <<
    \new Voice = "melody" \chant
    \new Lyrics \lyricsto "melody" \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \hide BarLine
    }
  }
}

```



Another common situation is transcription of neumatic or melismatic chants, i.e. chants with a varying number of notes to each syllable. In this case, one would want to set the syllable groups clearly apart, usually also the subdivisions of a longer melisma. One way to achieve this is to use a fixed `\time`, e.g., `1/4`, and let each syllable or note group fill one of these measures, with the help of tuplets or shorter durations. If the bar lines and all other rhythmical indications are made transparent, and the space around the bar lines is increased, this will give a fairly good representation in modern notation of the original.

To avoid that syllables of different width (such as “-ri” and “-rum”) spread the syllable note groups unevenly apart, the `'X-extent` property of the `LyricText` object may be set to a fixed value. Another, more cumbersome way would be to add the syllables as `\markup` elements. If further adjustments are necessary, this can be easily done with `s` ‘notes’.

```

spiritus = \relative {
  \time 1/4
  \override Lyrics.LyricText.X-extent = #'(0 . 3)
  d'4 \tuplet 3/2 { f8 a g } g a a4 g f8 e
  d4 f8 g g8 d f g a g f4 g8 a a4 s
  \tuplet 3/2 { g8 f d } e f g a g4
}

```

```

}

spirLyr = \lyricmode {
  Spi -- ri -- _ _ tus _ Do -- mi -- ni _ re -- ple -- _ vit _
  or -- _ bem _ ter -- ra -- _ rum, al -- _ _ le -- _ lu
  -- _ ia.
}
\score {
  \new Staff <<
    \new Voice = "chant" \spiritus
    \new Lyrics = "one" \lyricsto "chant" \spirLyr
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \override BarLine.X-extent = #'(-1 . 1)
      \hide Stem
      \hide Beam
      \hide BarLine
      \hide TupletNumber
    }
  }
}

```

10

## Ancient and modern from one source

*Using tags to produce mensural and modern music from the same source*

By using tags, it's possible to use the same music to produce both mensural and modern music. In this snippet, a function `menrest` is introduced, allowing mensural rests to be pitched as in the original, but with modern rests in the standard staff position. Tags are used to produce different types of bar line at the end of the music, but tags can also be used where other differences are needed: for example using “whole measure rests” (`R1`, `R\breve` etc.) in modern music, but normal rests (`r1`, `r\breve`, etc.) in the mensural version. Note that converting mensural music to its modern equivalent is usually referred to as **transcription**.

```

menrest = #(define-music-function (note)
  (ly:music?)
  #{
    \tag #'mens $(make-music 'RestEvent note)
    \tag #'mod $(make-music 'RestEvent note 'pitch '())
  })

```

```

MensStyle = {
  \autoBeamOff
  \override NoteHead #'style = #'petrucci
  \override Score.BarNumber #'transparent = ##t
  \override Stem.neutral-direction = #up
}

finalis = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::finalis
  \once \override BreathingSign.Y-offset = #0
  \once \override BreathingSign.minimum-X-extent = #'(-1.0 . 0.0)
  \once \override BreathingSign.minimum-Y-extent = #'(-2.5 . 2.5)

  \breathe
}

Music = \relative c'' {
  \set Score.tempoHideNote = ##t
  \key f \major
  \time 4/4
  g1 d'2 \menrest bes4 bes2 a2 r4 g4 fis2.
  \tag #'mens { \finalis }
  \tag #'mod { \bar "||" }
}

MenLyr = \lyricmode { So farre, deere life, deare life }
ModLyr = \lyricmode { So far, dear life, dear life }

\score {
  \keepWithTag #'mens {
    <<
    \new MensuralStaff
    {
      \new MensuralVoice = Cantus \clef "mensural-c1" \MensStyle \Music
    }
    \new Lyrics \lyricsto Cantus \MenLyr
  }
  >>
}

\score {
  \keepWithTag #'mod {
    \new ChoirStaff <<
    \new Staff
    {
      \new Voice = Sop \with {
        \remove "Note_heads_engraver"
        \consists "Completion_heads_engraver"
        \remove "Rest_engraver"
        \consists "Completion_rest_engraver" }
    {
      \shiftDurations #1 #0 { \autoBeamOff \Music }
    }
  }
  >>
}

```

```

    }
  }
  \new Lyrics \lyricsto Sop \ModLyr
  >>
}
}

```



## Editorial markings

### 2.10 World music

The purpose of this section is to highlight musical notation issues that are relevant to traditions outside the Western tradition.

#### 2.10.1 Common notation for non-Western music

This section discusses how to enter and print music scores that do not belong to the Western classical tradition, also referred to as *Common Practice Period*.

### Extending notation and tuning systems

Standard classical notation (also known as *Common Practice Period* notation) is commonly used in all sorts of music, not limited to ‘classical’ Western music. This notation is discussed in [\[Writing pitches\]](#), pagina [\[undefined\]](#), and the various note names that may be used are explained in [\[Note names in other languages\]](#), pagina [\[undefined\]](#).

However, many types of non-Western music (and some types of Western folk and traditional music) employ alternative or extended tuning systems that do not fit readily into standard classical notation.

In some cases standard notation is still used, with the pitch differences being implicit. For example, *Arabic music* is notated with standard semitone and quarter-tone accidentals, with the precise pitch alterations being determined by context. Italian note names are typically used, while the init file `arabic.ly` provides a suitable set of macros and definitions extending the standard notation. For more details, see Sezione 2.10.2 [\[Arabic music\]](#), pagina 464.

Other types of music require extended or unique notations. *Turkish classical music* or Ottoman music, for example, employs melodic forms known as *makamlar*, whose intervals are based on 1/9 divisions of the whole tone. Standard Western staff notes are still used, but with special accidentals unique to Turkish music, that are defined in the file `makam.ly`. For further information on Turkish classical music and makamlar, see Sezione 2.10.3 [\[Turkish classical music\]](#), pagina 469.

To locate init files such as `arabic.ly` or `makam.ly` on your system, see Sezione “Other sources of information” in *Manuale di Apprendimento*.



## Frammenti di codice selezionati

### *Makam example*

Makam is a type of melody from Turkey using 1/9th-tone microtonal alterations. Consult the initialization file ‘ly/makam.ly’ for details of pitch names and alterations.

```
% Initialize makam settings
\include "makam.ly"

\relative c' {
  \set Staff.keyAlterations = #`((6 . ,(- KOMA)) (3 . ,BAKIYE))
  c4 cc db fk
  gbm4 gfc gfb efk
  fk4 db cc c
}
```



## Vedi anche

Music Glossary: Sezione “Common Practice Period” in *Glossario Musicale*, Sezione “makam-lar” in *Glossario Musicale*.

Learning Manual: Sezione “Other sources of information” in *Manuale di Apprendimento*.

Notation Reference: <undefined> [Writing pitches], pagina <undefined>, <undefined> [Note names in other languages], pagina <undefined>, Sezione 2.10.2 [Arabic music], pagina 464, Sezione 2.10.3 [Turkish classical music], pagina 469.

## 2.10.2 Arabic music

This section highlights issues that are relevant to notating Arabic music.

### References for Arabic music

Arabic music so far has been mainly an oral tradition. When music is transcribed, it is usually in a sketch format, on which performers are expected to improvise significantly. Increasingly, Western notation, with a few variations, is adopted in order to communicate and preserve Arabic music.

Some elements of Western musical notation such as the transcription of chords or independent parts, are not required to typeset the more traditional Arabic pieces. There are however some different issues, such as the need to indicate medium intervals that are somewhere between a semi-tone and a tone, in addition to the minor and major intervals that are used in Western music. There is also the need to group and indicate a large number of different maqams (modes) that are part of Arabic music.

In general, Arabic music notation does not attempt to precisely indicate microtonal elements that are present in musical practice.

Several issues that are relevant to Arabic music are covered elsewhere:

- Note names and accidentals (including quarter tones) can be tailored as discussed in Sezione 2.10.1 [Common notation for non-Western music], pagina 463.
- Additional key signatures can also be tailored as described in <undefined> [Key signature], pagina <undefined>.

- Complex time signatures may require that notes be grouped manually as described in [\(undefined\)](#) [Manual beams], pagina [\(undefined\)](#).
- *Takasim* which are rhythmically free improvisations may be written down omitting bar lines as described in [\(undefined\)](#) [Unmetered music], pagina [\(undefined\)](#).

## Vedi anche

Notation Reference: Sezione 2.10.1 [Common notation for non-Western music], pagina 463, [\(undefined\)](#) [Key signature], pagina [\(undefined\)](#), [\(undefined\)](#) [Manual beams], pagina [\(undefined\)](#).

Snippets: Sezione “World music” in *Frammenti di codice*.

## Arabic note names

The more traditional Arabic note names can be quite long and are not suitable for the purpose of music writing, so they are not used. English note names are not very familiar in Arabic music education, so Italian or Solfege note names (**do, re, mi, fa, sol, la, si**) are used instead; modifiers (accidentals) can also be used. Italian note names and accidentals are explained in [\(undefined\)](#) [Note names in other languages], pagina [\(undefined\)](#); the use of standard Western notation to notate non-Western music is discussed in Sezione 2.10.1 [Common notation for non-Western music], pagina 463.

For example, this is how the Arabic *rast* scale can be notated:

```
\include "arabic.ly"
\relative {
  do' re misb fa sol la sisb do sisb la sol fa misb re do
}
```



The symbol for semi-flat does not match the symbol which is used in Arabic notation. The `\down` symbol defined in `arabic.ly` may be used preceding a flat symbol as a work around if it is important to use the specific Arabic semi-flat symbol. The appearance of the semi-flat symbol in the key signature cannot be altered by using this method.

```
\include "arabic.ly"
\relative {
  \set Staff.extraNatural = ##f
  dod' dob dosd \down dob dobsd dodsd do do
}
```



## Vedi anche

Notation Reference: [\(undefined\)](#) [Note names in other languages], pagina [\(undefined\)](#), Sezione 2.10.1 [Common notation for non-Western music], pagina 463.

Snippets: Sezione “World music” in *Frammenti di codice*.

## Arabic key signatures

In addition to the minor and major key signatures, the following key signatures are defined in `arabic.ly`: *bayati*, *rast*, *sikah*, *iraq*, and *kurd*. These key signatures define a small number of maqam groups rather than the large number of maqams that are in common use.

In general, a maqam uses the key signature of its group, or a neighbouring group, and varying accidentals are marked throughout the music.

For example to indicate the key signature of a maqam muhayer piece:

```
\key re \bayati
```

Here *re* is the default pitch of the muhayer maqam, and *bayati* is the name of the base maqam in the group.

While the key signature indicates the group, it is common for the title to indicate the more specific maqam, so in this example, the name of maqam muhayer should appear in the title.

Other maqams in the same bayati group, as shown in the table below: (bayati, hussaini, saba, and ushaq) can be indicated in the same way. These are all variations of the base and most common maqam in the group, which is bayati. They usually differ from the base maqam in their upper tetrachords, or certain flow details that don't change their fundamental nature, as siblings.

The other maqam in the same group (Nawa) is related to bayati by modulation which is indicated in the table in parenthesis for those maqams that are modulations of their base maqam. Arabic maqams admit of only limited modulations, due to the nature of Arabic musical instruments. Nawa can be indicated as follows:

```
\key sol \bayati
```

In Arabic music, the same term such as bayati that is used to indicate a maqam group, is also a maqam which is usually the most important in the group, and can also be thought of as a base maqam.

Here is one suggested grouping that maps the more common maqams to key signatures:

maqam group	key	finalis	Other maqmas in group (finalis)
ajam	major	sib	jaharka (fa)
bayati	bayati	re	hussaini, muhayer, saba, ushaq, nawa (sol)
hijaz	kurd	re	shahnaz, shad arban (sol), hijazkar (do)
iraq	iraq	sisb	-
kurd	kurd	re	hijazkar kurd (do)
nahawand	minor	do	busalik (re), farah faza (sol)
nakriz	minor	do	nawa athar, hisar (re)
rast	rast	do	mahur, yakah (sol)
sikah	sikah	misb	huzam

## Frammenti di codice selezionati

*Armature di chiave non tradizionali*

Il comando `\key` comunemente usato imposta la proprietà `keyAlterations`, che fa parte del contesto `Staff`.

Per creare armature di chiave non standard, tale proprietà va impostata esplicitamente. Il formato di questo comando è una lista:

```
\set Staff.keyAlterations = #`(((ottava . grado) . alterazione) ((ottava .
grado) . alterazione) ...) dove, per ogni elemento della lista, ottava indica l'ottava
(0 è l'ottava dal Do centrale al Si precedente), grado indica la nota all'interno dell'ottava
(0 significa Do e 6 significa Si) e alterazione può essere ,SHARP ,FLAT ,DOUBLE-SHARP etc.
(Si noti la virgola iniziale.)
```

Altrimenti, usando, per ogni elemento della lista, il formato breve (**grado . alterazione**), ciò indica che la stessa alterazione deve essere presente in tutte le ottave.

Ecco un esempio di una possibile armatura per generare una scala a tono intero:

```
\relative {
  \set Staff.keyAlterations = #`((6 . ,FLAT)
                                (5 . ,FLAT)
                                (3 . ,SHARP))

  c'4 d e fis
  aes4 bes c2
}
```



## Vedi anche

Music Glossary: Sezione “maqam” in *Glossario Musicale*, Sezione “bayati” in *Glossario Musicale*, Sezione “rast” in *Glossario Musicale*, Sezione “sikah” in *Glossario Musicale*, Sezione “iraq” in *Glossario Musicale*, Sezione “kurd” in *Glossario Musicale*.

Notation Reference: [\[Key signature\]](#), pagina [\[undefined\]](#).

Learning Manual: Sezione “Pitches and key signatures” in *Manuale di Apprendimento*.

Internals Reference: Sezione “KeySignature” in *Guida al Funzionamento Interno*.

Snippets: Sezione “World music” in *Frammenti di codice*, Sezione “Pitches” in *Frammenti di codice*.

## Arabic time signatures

Some Arabic and Turkish music classical forms such as *Semai* use unusual time signatures such as 10/8. This may lead to an automatic grouping of notes that is quite different from existing typeset music, where notes may not be grouped on the beat, but in a manner that is difficult to match by adjusting automatic beaming. The alternative is to switch off automatic beaming and beam the notes manually. Even if a match to existing typeset music is not required, it may still be desirable to adjust the automatic beaming behaviour and/or use compound time signatures.

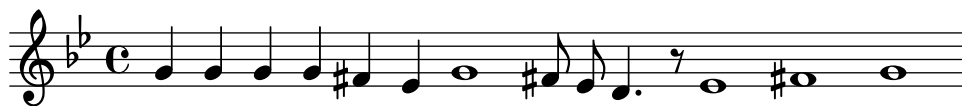
## Frammenti di codice selezionati

### *Arabic improvisation*

For improvisations or taqasim which are temporarily free, the time signature can be omitted and `\cadenzaOn` can be used. Adjusting the accidental style might be required, since the absence of bar lines will cause the accidental to be marked only once. Here is an example of what could be the start of a hijaz improvisation:

```
\include "arabic.ly"

\relative sol' {
  \key re \kurd
  \accidentalStyle forget
  \cadenzaOn
  sol4 sol sol sol fad mib sol1 fad8 mib re4. r8 mib1 fad sol
}
```



## Vedi anche

Music Glossary: Sezione “semai” in *Glossario Musicale*, Sezione “taqasim” in *Glossario Musicale*.

Notation Reference: [\[Manual beams\]](#), pagina [\[Automatic beams\]](#), pagina [\[Unmetered music\]](#), pagina [\[Automatic accidentals\]](#), pagina [\[Setting automatic beam behavior\]](#), pagina [\[Time signature\]](#), pagina [\[Time signature\]](#).

Snippets: Sezione “World music” in *Frammenti di codice*.

## Arabic music example

Here is a template that also uses the start of a Turkish *Semai* that is familiar in Arabic music education in order to illustrate some of the peculiarities of Arabic music notation, such as medium intervals and unusual modes that are discussed in this section.

```
\include "arabic.ly"
\score {
  \relative {
    \set Staff.extraNatural = ##f
    \set Staff.autoBeaming = ##f
    \key re \bayati
    \time 10/8

    re'4 re'8 re16 [misb re do] sisb [la sisb do] re4 r8
    re16 [misb do re] sisb [do] la [sisb sol8] la [sisb] do [re] misb
    fa4 fa16 [misb] misb8. [re16] re8 [misb] re [do] sisb
    do4 sisb8 misb16 [re do sisb] la [do sisb la] la4 r8
  }
  \header {
    title = "Semai Muhayer"
    composer = "Jamil Bek"
  }
}
```



## Vedi anche

Snippets: Sezione “World music” in *Frammenti di codice*.

## Further reading for Arabic music

1. *The music of the Arabs* by Habib Hassan Touma [Amadeus Press, 1996], contains a discussion of maqams and their method of groupings.

There are also various web sites that explain maqams and some provide audio examples such as :

- <http://www.maqamworld.com/>
- <http://www.turath.org/>

There are some variations in the details of how maqams are grouped, despite agreement on the criteria of grouping maqams that are related through common lower tetra chords, or through modulation.

2. There is not a complete consistency, sometimes even in the same text on how key signatures for particular maqams should be specified. It is common, however, to use a key signature per group, rather than a different key signature for each different makam.

Method books by the following authors for the *Oud*, the Arabic lute, contain examples of mainly Turkish and Arabic compositions.

- Charbel Rouhana
- George Farah
- Ibrahim Ali Darwish Al-masri

### 2.10.3 Turkish classical music

This section highlights issues that are relevant to notating Turkish classical music.

#### References for Turkish classical music

Turkish classical music developed in the Ottoman Empire in a period roughly contemporaneous with classical music in Europe, and has continued on into the 20th and 21st centuries as a vibrant and distinct tradition with its own compositional forms, theory and performance styles. Among its striking features is the use of microtonal intervals based on ‘commas’ of  $1/9$  of a tone, from which are constructed the melodic forms known as *makam* (plural *makamlar*).

Some issues relevant to Turkish classical music are covered elsewhere:

- Special note names and accidentals are explained in Sezione 2.10.1 [Common notation for non-Western music], pagina 463.

#### Turkish note names

Pitches in Turkish classical music traditionally have unique names, and the basis of pitch on  $1/9$ -tone divisions means makamlar employ a completely different set of intervals from Western scales and modes: *koma* ( $1/9$  of a tone), *eksik bakiye* ( $3/9$ ), *bakiye* ( $4/9$ ), *küçük mücenneb* ( $5/9$ ), *büyük mücenneb* ( $8/9$ ), *tanîni* (a whole tone) and *artık ikili* ( $12/9$  or  $13/9$  of a tone).

From a modern notational point of view it is convenient to use the standard Western staff notes (c, d, e, ...) with special accidentals that raise or lower notes by intervals of  $1/9$ ,  $4/9$ ,  $5/9$  and  $8/9$  of a tone. These accidentals are defined in the file `makam.ly`.

The following table lists:

- the name of these special accidentals,
- the accidental suffix that must be added to notes,
- and their pitch alteration as a fraction of one whole tone.

Accidental name	suffix	pitch alteration
büyük mücenneb (sharp)	-bm	+ $8/9$
küçük mücenneb (sharp)	-k	+ $5/9$
bakiye (sharp)	-b	+ $4/9$

koma (sharp)	-c	+1/9
koma (flat)	-fc	-1/9
bakiye (flat)	-fb	-4/9
küçük mücenneb (flat)	-fk	-5/9
büyük mücenneb (flat)	-fbm	-8/9

For a more general explanation of non-Western music notation, see Sezione 2.10.1 [Common notation for non-Western music], pagina 463.

### Vedi anche

Music Glossary: Sezione “makam” in *Glossario Musicale*, Sezione “makamlar” in *Glossario Musicale*.

Notation Reference: Sezione 2.10.1 [Common notation for non-Western music], pagina 463.

## 3 Input e output

Questa sezione tratta le questioni generali relative all'input e all'output di LilyPond, non specifiche di un certo tipo di notazione.

### 3.1 Struttura dell'input

Il principale formato di input di LilyPond sono i file di testo. Per convenzione, questi file hanno estensione `.ly`.

#### 3.1.1 Struttura di una partitura

Un blocco `\score` deve contenere una singola espressione musicale delimitata da parentesi graffe:

```
\score {
  ...
}
```

**Nota:** Ci deve essere **solo una** espressione musicale più esterna in un blocco `\score` e **deve** essere racchiusa tra parentesi graffe.

Questa espressione musicale singola può essere di qualsiasi dimensione e contenere altre espressioni musicali di qualsiasi complessità. Tutti gli esempi seguenti sono espressioni musicali:

```
{ c'4 c' c' c' }
```

```
{
```

```
  { c'4 c' c' c' }
```

```
  { d'4 d' d' d' }
```

```
}
```

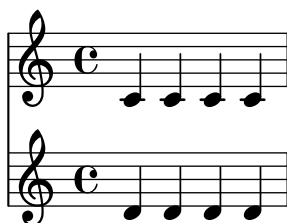


```
<<
```

```
\new Staff { c'4 c' c' c' }
```

```
\new Staff { d'4 d' d' d' }
```

```
>>
```



```
{
```

```
\new GrandStaff <<
```

```
  \new StaffGroup <<
```

```
    \new Staff { \flauto }
```

```
    \new Staff { \oboe }
```

```
  >>
```

```
  \new StaffGroup <<
```

```
    \new Staff { \violinoI }
```



```

\new Staff { \violinoII }
>>
>>
}

```

I commenti sono un'eccezione a questa regola generale (altre eccezioni sono spiegate in [\[File structure\]](#), pagina [\[undefined\]](#)). Sia i commenti su una singola riga che quelli multiriga delimitati da `%{ ... %}` possono essere inseriti ovunque nel file di input: dentro o fuori un blocco `\score` e dentro o fuori la singola espressione musicale di un blocco `\score`.

È bene ricordare che anche se un file contiene soltanto un blocco `\score`, questo è implicitamente racchiuso in un blocco `\book`. Un blocco `\book` in un file sorgente produce almeno un file di output e il nome predefinito del file di output deriva dal nome del file di input, quindi `fandangoperelefanti.ly` genererà `fandangoperelefanti.pdf`.

Maggiori dettagli sui blocchi `\book` si trovano in [\[Multiple scores in a book\]](#), pagina [\[undefined\]](#), [\[Multiple output files from one input file\]](#), pagina [\[undefined\]](#), e [\[File structure\]](#), pagina [\[undefined\]](#).

## Vedi anche

Manuale di apprendimento: Sezione “Lavorare sui file di input” in *Manuale di Apprendimento*, Sezione “Espressioni musicali” in *Manuale di Apprendimento*, Sezione “La partitura è una (singola) espressione musicale composta” in *Manuale di Apprendimento*.

### 3.1.2 Molteplici partiture in un libro

Un documento può contenere più brani di musica e testo, come, per esempio, uno studio o una parte orchestrale con vari movimenti. Ogni movimento si inserisce con un blocco `\score`,

```

\score {
  ...musica...
}

```

e il testo si inserisce con un blocco `\markup`,

```

\markup {
  ...testo...
}

```

Tutti i movimenti e i testi che appaiono nello stesso file `.ly` normalmente vengono elaborati in un singolo file di output.

```

\score {
  ...
}
\markup {
  ...
}
\score {
  ...
}

```

Un'importante eccezione è costituita dai documenti da elaborare con `lilypond-book`, dove occorre aggiungere esplicitamente un blocco `\book`, altrimenti apparirà nell'output solo il primo blocco `\score` o `\markup`.

L'intestazione di ogni brano musicale può essere inserita nel blocco `\score`. Il nome definito nel campo `piece` (brano) dell'intestazione apparirà all'inizio di ogni movimento. Il titolo dell'intero libro può trovarsi all'interno del blocco `\book` oppure, se questo non è presente, nel blocco `\header` all'inizio del file.

```

\header {

```

```

    title = "Otto miniature"
    composer = "Igor Stravinsky"
}
\score {
    ...
    \header { piece = "Romanza" }
}
\markup {
    ...testo della seconda strofa...
}
\markup {
    ...testo della terza strofa...
}
\score {
    ...
    \header { piece = "Minuetto" }
}

```

I brani musicali possono essere raggruppati in parti di libro tramite i blocchi `\bookpart`. Le parti di libro sono separate da un'interruzione di pagina e possono iniziare con un titolo, come il libro stesso, specificandolo in un blocco `\header`.

```

\bookpart {
    \header {
        title = "Titolo del libro"
        subtitle = "Prima parte"
    }
    \score { ... }
    ...
}
\bookpart {
    \header {
        subtitle = "Seconda parte"
    }
    \score { ... }
    ...
}

```

### 3.1.3 Molteplici file di output da un unico file di input

Per generare molteplici file di output dallo stesso file `.ly`, basta aggiungere molteplici blocchi `\book`, ognuno dei quali produrrà un file di output separato. Se non è specificato alcun blocco `\book` nel file di input, LilyPond tratterà implicitamente l'intero file come un singolo blocco `\book`, come è spiegato in [\[File structure\]](#), pagina [\[File structure\]](#).

Nel generare molteplici file da un singolo file sorgente, Lilypond controlla che nessuno dei file di output di alcun blocco `\book` sovrascriva il file di output prodotto da un blocco `\book` precedente dello stesso file di input.

Per farlo, aggiunge un suffisso al nome del file di output di ogni blocco `\book`, derivato dal nome del file di input (se viene lasciata l'impostazione predefinita).

Il comportamento predefinito consiste quindi nell'appendere un suffisso numerico a ogni nome che potrebbe entrare in conflitto, quindi

```

\book {
    \score { ... }
}

```

```

\paper { ... }
}
\book {
  \score { ... }
  \paper { ... }
}
\book {
  \score { ... }
  \paper { ... }
}

```

nel file sorgente `ottominiature.ly` genererà

- `ottominiature.pdf`,
- `ottominiature-1.pdf` e
- `ottominiature-2.pdf`.

### 3.1.4 Nomi dei file di output

Lilypond permette di decidere quali nomi di file debbano essere usati dai vari backend quando questi generano i file di output.

Nella sezione precedente abbiamo visto come Lilypond prevenga i conflitti di nome quando genera molti file di output da un singolo file sorgente. È possibile anche specificare un proprio suffisso per ogni blocco `\book`. Quindi si possono, per esempio, generare file chiamati `ottominiature-Romanza.pdf`, `ottominiature-Minuetto.pdf` e `ottominiature-Notturmo.pdf` aggiungendo una dichiarazione `\bookOutputSuffix` all'interno di ogni blocco `\book`.

```

\book {
  \bookOutputSuffix "Romanza"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputSuffix "Minuetto"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputSuffix "Notturmo"
  \score { ... }
  \paper { ... }
}

```

È possibile anche specificare un diverso nome del file di output per ciascun blocco `book`, tramite le dichiarazioni `\bookOutputName`

```

\book {
  \bookOutputName "Romanza"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputName "Minuetto"
  \score { ... }
  \paper { ... }
}

```

```

}
\book {
  \bookOutputName "Notturmo"
  \score { ... }
  \paper { ... }
}

```

Questo file produrrà i seguenti file di output:

- Romanza.pdf,
- Minuetto.pdf e
- Notturmo.pdf.

### 3.1.5 Struttura del file

Un file `.ly` può contenere un qualsiasi numero di espressioni di livello superiore (in inglese, *toplevel expressions*). Per espressione di livello superiore si intende una delle seguenti:

- Una definizione di output, come `\paper`, `\midi` e `\layout`. Tale definizione, se posta nel livello superiore, cambia le impostazioni predefinite al livello del libro. Se più di una di queste definizioni viene inserita nel livello superiore, le definizioni vengono combinate, ma in caso di conflitto hanno precedenza le definizioni più recenti. Per sapere con precisione come ciò influisca sul blocco `\layout`, leggere `<undefined>` [Il blocco `\layout`], pagina `<undefined>`.
- Un'espressione scheme diretta, come  `#(set-default-paper-size "a7" 'landscape)` o  `#(ly:set-option 'point-and-click #f)`.
- Un blocco `\header`. Se all'inizio del file, imposta il blocco dell'intestazione globale. Questo è il blocco che contiene le impostazioni predefinite dei campi dei titoli come compositore, titolo, etc. per tutti i libri del file (vedi `<undefined>` [Titles explained], pagina `<undefined>`).
- Un blocco `\score`. Questa partitura e altre eventuali partiture di livello superiore saranno combinate insieme in un singolo blocco `\book`. Tale comportamento può essere modificato impostando la variabile `toplevel-score-handler` nel livello superiore. Il gestore (in inglese *handler*) predefinito è definito nel file di inizializzazione `../scm/lily.scm`.
- Un blocco `\book` combina logicamente molteplici movimenti (ovvero molteplici blocchi `\score`) in un documento. Se ci sono vari blocchi `\score`, verrà creato un file di output per ogni blocco `\book`, in cui saranno concatenati tutti i movimenti corrispondenti. Ha senso specificare esplicitamente i blocchi `\book` in un file `.ly` solo se si desidera creare vari file di output da un solo file di input. Un'eccezione è data dai documenti `lilypond-book`, dove bisogna aggiungere esplicitamente un blocco `\book` se si vuole più di un singolo blocco `\score` o `\markup` nello stesso esempio. Tale comportamento può essere modificato impostando la variabile `toplevel-book-handler` nel livello superiore. Il gestore predefinito è definito nel file di inizializzazione `../scm/lily.scm`.
- Un blocco `\bookpart`. Un libro può essere suddiviso in varie parti, tramite blocchi `\bookpart`, per semplificare le interruzioni di pagina o per usare impostazioni `\paper` diverse nelle varie parti.
- Un'espressione musicale composta, come

```
{ c'4 d' e'2 }
```

pone il brano in un blocco `\score` e lo formatta in un unico libro insieme a tutti gli altri blocchi `\score` e espressioni musicali di livello superiore. In altre parole, un file che contiene soltanto l'espressione musicale precedente verrà trasformato in

```

\book {
  \score {
    \new Staff {
      \new Voice {

```

```

        { c'4 d' e'2 }
      }
    }
  \layout { }
}
\paper { }
\header { }
}

```

Tale comportamento può essere modificato impostando la variabile `toplevel-music-handler` nel livello superiore. Il gestore predefinito è definito nel file di inizializzazione `../scm/lily.scm`.

- Un testo, per esempio una strofa

```

\markup {
  2. La prima riga della seconda strofa.
}

```

I testi possono trovarsi sopra, sotto o in mezzo alle partiture o espressioni musicali, ovunque esse appaiano.

- Una variabile, come

```
foo = { c4 d e d }
```

può essere utilizzata in un punto successivo del file scrivendo `\foo`. Il nome di una variabile deve avere solo caratteri alfabetici; nessun numero, trattino o trattino basso.

L'esempio seguente mostra tre elementi che possono essere inseriti nel livello superiore

```

\layout {
  % Non giustificare l'output
  ragged-right = ##t
}

\header {
  title = "Do-re-mi"
}

```

```
{ c'4 d' e2 }
```

Ciascuna delle seguenti istruzioni lessicali può essere inserita in qualsiasi punto di un file:

- `\version`
- `\include`
- `\sourcefilename`
- `\sourcefileline`
- Un commento su riga singola, introdotto da un segno `%`.
- Un commento multiriga delimitato da `%{ ... %}`.

Lo spazio bianco tra gli elementi dell'input viene generalmente ignorato e può essere liberamente omesso o aumentato per migliorare la leggibilità. Tuttavia esistono dei casi in cui lo spazio bianco deve essere sempre usato per non incorrere in errori:

- Intorno ad ogni parentesi graffa di apertura e di chiusura.
- Dopo ogni comando o variabile, ovvero qualsiasi elemento che inizi con un segno `\`.
- Dopo ogni elemento che debba essere interpretato come un'espressione Scheme, ovvero ogni elemento che inizi con un segno `#`.
- Per separare tutti gli elementi di un'espressione Scheme.
- Nella modalità `lyricmode` prima e dopo i comandi `\set` e `\override`.

## Vedi anche

Manuale di apprendimento: Sezione “Come funzionano i file di input di LilyPond” in *Manuale di Apprendimento*.

Guida alla notazione: [\[Titles explained\]](#), pagina [\[Il blocco \layout\]](#), pagina [\[Il blocco \layout\]](#).

## 3.2 Titoli e intestazioni

Quasi tutte le partiture musicali hanno il titolo e il nome del compositore e alcuni brani comprendono molte altre informazioni.

### 3.2.1 Creazione di titoli intestazioni e piè di pagina

#### Come funzionano i titoli

Ogni blocco `\book` in un singolo file di input produce un diverso file di output, vedi [\[File structure\]](#), pagina [\[Default layout of bookpart and score titles\]](#). In ciascun file di output sono disponibili tre tipi di titolazioni: *titoli del libro* all’inizio di ogni libro (*book*), *titoli della parte* all’inizio di ciascuna parte (*bookpart*) e *titoli del brano* all’inizio di ciascun brano (*score*).

I valori dei campi dei titoli come `title` (titolo) e `composer` (compositore) sono definiti nei blocchi `\header` (la sintassi dei blocchi `\header` e un elenco completo dei campi disponibili si trovano in [\[Default layout of bookpart and score titles\]](#), pagina [\[Default layout of bookpart and score titles\]](#)). I titoli del libro, delle parti e dei brani possono avere tutti gli stessi campi, sebbene per impostazione predefinita i campi dei titoli del brano siano limitati a `piece` e `opus`.

I blocchi `\header` possono essere inseriti in quattro diversi punti formando una gerarchia discendente di blocchi `\header`:

- All’inizio del file di input, prima di tutti i blocchi `\book`, `\bookpart` e `\score`.
- All’interno di un blocco `\book` ma fuori da tutti i blocchi `\bookpart` e `\score` compresi in quel libro.
- All’interno di un blocco `\bookpart` ma fuori da tutti i blocchi `\score` compresi in quella parte.
- Dopo l’espressione musicale in un blocco `\score`.

I valori dei campi vengono filtrati attraverso questa gerarchia, con i valori più in alto nella gerarchia che persistono finché un valore più in basso nella gerarchia non ha la precedenza. In sintesi:

- Il titolo di un libro deriva dai campi impostati all’inizio del file di input, modificati dai campi definiti nel blocco `\book`. I valori risultanti vengono usati per stampare il titolo di quel libro, purché ci sia altro materiale che generi una pagina all’inizio del libro, prima della prima parte. Una singola interruzione di pagina (`\pageBreak`) è sufficiente.
- Il titolo di una parte deriva dai campi impostati all’inizio del file di input, modificati dai campi definiti nel blocco `\book` e poi da quelli definiti nel blocco `\bookpart`. I valori risultanti vengono usati per stampare il titolo di quella parte del libro.
- Il titolo di un brano deriva dai campi all’inizio del file di input, modificati dai campi definiti nel blocco `\book` e poi da quelli definiti nel blocco `\bookpart` e infine da quelli definiti nel blocco `\score`. I valori risultanti vengono usati per stampare il titolo di quel brano. Nota bene: per impostazione predefinita nei titoli del brano appaiono soltanto i campi `piece` e `opus` a meno che la variabile `print-all-headers` di `\paper` non sia impostata su `#t`.

**Nota:** Ricordarsi che quando si mette un blocco `\header` in un blocco `\score`, l’espressione musicale deve precedere il blocco `\header`.

Non è necessario inserire blocchi `\header` in tutti e quattro i luoghi: alcuni o perfino tutti possono essere omessi. Analogamente, in semplici file di input si possono omettere i blocchi `\book` e `\bookpart`, lasciando che questi siano creati implicitamente.

Se il libro ha un solo brano, il blocco `\header` viene di solito messo all'inizio del file, in modo che sia prodotto soltanto il titolo della parte e lasciando disponibili tutti i campi di titolazione.

Se il libro ha vari brani, è possibile usare diverse disposizioni dei blocchi `\header`, corrispondenti ai diversi tipi di pubblicazione musicale. Per esempio, se la pubblicazione contiene vari brani dello stesso compositore, la soluzione più adatta prevederebbe un blocco `\header` all'inizio del file che specifichi il titolo del libro e il nome del compositore, e dei blocchi `\header` in ciascun blocco `\score` che specifichino il titolo del brano (*piece*) e dell'opera (*opus*), come in questo esempio:

```
\header {
  title = "SUITE I."
  composer = "J. S. Bach."
}

\score {
  \new Staff \relative {
    \clef bass
    \key g \major
    \repeat unfold 2 { g,16( d' b') a b d, b' d, } |
    \repeat unfold 2 { g,16( e' c') b c e, c' e, } |
  }
  \header {
    piece = "Prélude."
  }
}

\score {
  \new Staff \relative {
    \clef bass
    \key g \major
    \partial 16 b16 |
    <g, d' b'~>4 b'16 a( g fis) g( d e fis) g( a b c) |
    d16( b g fis) g( e d c) b(c d e) fis( g a b) |
  }
  \header {
    piece = "Allemande."
  }
}
```

## SUITE I.

J. S. Bach.

Prélude.



Allemande.



Sono possibili disposizioni più complesse. Per esempio, i campi testuali del blocco `\header` di un libro possono essere stampati nei titoli di tutti i brani, magari sovrascrivendo alcuni campi e sopprimendone altri:

```
\book {
  \paper {
    print-all-headers = ##t
  }
  \header {
    title = "DAS WOHLTEMPERIRTE CLAVIER"
    subtitle = "TEIL I"
    % Non mostrare il piè di pagina predefinito nell'ultima pagina di questo libro
    tagline = ##f
  }
  \markup { \vspace #1 }
  \score {
    \new PianoStaff <<
      \new Staff { s1 }
      \new Staff { \clef "bass" s1 }
    >>
    \header {
      title = "PRAELUDIUM I"
      opus = "BWV 846"
      % Non mostrare il sottotitolo in questo brano
      subtitle = ##f
    }
  }
  \score {
    \new PianoStaff <<
      \new Staff { s1 }
      \new Staff { \clef "bass" s1 }
    >>
    \header {
      title = "FUGA I"
      subsubtitle = "A 4 VOCI"
      opus = "BWV 846"
      % Non mostrare il sottotitolo in questo brano
      subtitle = ##f
    }
  }
}
```

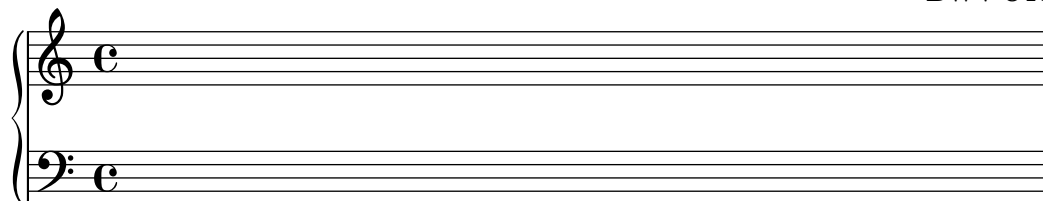


# DAS WOHLTEMPERIRTE CLAVIER

## TEIL I

### PRAELUDIUM I

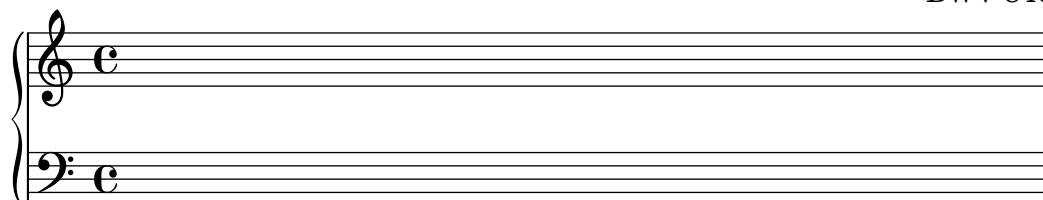
BWV 846



### FUGA I

A 4 VOCI

BWV 846



#### Vedi anche

Guida alla notazione: [\[File structure\]](#), pagina [\[Default layout of bookpart and score titles\]](#), pagina [\[Custom layout for titles\]](#), pagina [\[Custom layout for titles\]](#).

#### Formattazione predefinita dei titoli delle parti e dei brani

Questo esempio illustra visivamente tutte le variabili del blocco `\header`:

```
\book {
  \header {
    % I seguenti campi sono centrati
    dedication = "Dedica"
    title = "Titolo"
    subtitle = "Sottotitolo"
    subsubtitle = "Sottosottotitolo"
    % I seguenti campi sono distribuiti uniformemente su una riga
    % il campo "instrument" appare anche nelle pagine seguenti
    instrument = \markup \with-color #green "Strumento"
    poet = "Poeta"
    composer = "Compositore"
    % I campi seguenti sono posti agli estremi opposti della stessa riga
    meter = "Tempo"
    arranger = "Arrangiatore"
    % I campi seguenti sono centrati in fondo
    tagline = "Lo slogan va in fondo all'ultima pagina"
    copyright = "Il copyright va in fondo alla prima pagina"
  }
}
```

```

\score {
  { s1 }
  \header {
    % I campi seguenti sono posti agli estremi opposti della stessa riga
    piece = "Brano 1"
    opus = "Opera 1"
  }
}
\score {
  { s1 }
  \header {
    % I campi seguenti sono posti agli estremi opposti della stessa riga
    piece = "Brano 2 sulla stessa pagina"
    opus = "Opera 2"
  }
}
\pageBreak
\score {
  { s1 }
  \header {
    % I campi seguenti sono posti agli estremi opposti della stessa riga
    piece = "Brano 3 su una nuova pagina"
    opus = "Opera 3"
  }
}
}

```

Dedica

**Titolo****Sottotitolo****Sottosottotitolo****Strumento**

Poeta

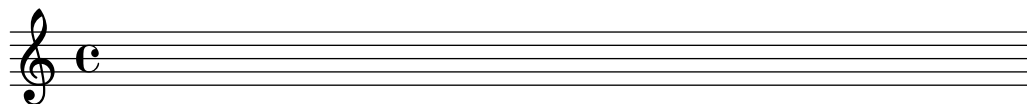
Compositore

Tempo

Arrangiatore

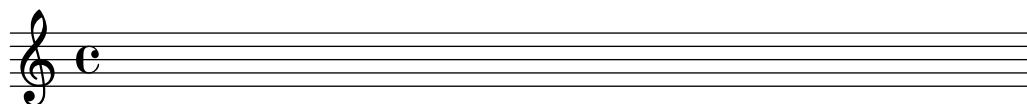
Brano 1

Opera 1



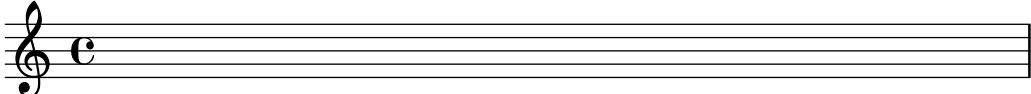
Brano 2 sulla stessa pagina

Opera 2



Il copyright va in fondo alla prima pagina

2	Strumento	
Brano 3 su una nuova pagina		Opera 3



### Lo slogan va in fondo all'ultima pagina

Notare che:

- Il nome dello strumento sarà ripetuto su ogni pagina.
- Appaiono soltanto i campi `piece` e `opus` di un blocco `\score` quando la variabile `print-all-headers` del foglio è impostata su `##f` (valore predefinito).
- I campi testuali non specificati in un blocco `\header` sono sostituiti con `\null` in modo da non sprecare spazio.
- Le impostazioni predefinite per `scoreTitleMarkup` posizionano i campi `piece` e `opus` alle estremità opposte della stessa riga.

Per cambiare la formattazione predefinita leggere `<undefined> [Custom layout for titles]`, pagina `<undefined>`.

Se un blocco `\book` inizia subito con un blocco `\bookpart`, non verrà stampato alcun titolo per il libro, dato che non esiste una pagina in cui farlo apparire. Se il titolo del libro deve comunque apparire, conviene inserire all'inizio di `\book` del testo inserito con `\markup` oppure un comando `\pageBreak`.

Usare la variabile `breakbefore` all'interno di un blocco `\header` racchiuso in un blocco `\score` per far sì che i titoli del blocco `\header` di più alto livello appaiano da soli nella prima pagina, mentre la musica (definita nel blocco `\score`) inizia nella pagina successiva.

```
\book {
  \header {
    title = "Titolo del libro"
    subtitle = "Sottotitolo del libro"
    copyright = "Fine della prima pagina"
  }
  \score {
    \repeat unfold 4 { e'' e'' e'' e'' }
    \header {
      piece = "Titolo del brano"
      breakbefore = ##t
    }
  }
}
```

}

## Titolo del libro

### Sottotitolo del libro

Fine della prima pagina

2

Titolo del brano



Music engraving by LilyPond 2.19.36—[www.lilypond.org](http://www.lilypond.org)

### Vedi anche

Manuale di apprendimento: Sezione “Come funzionano i file di input di LilyPond” in *Manuale di Apprendimento*,

Guida alla notazione: [\[Custom layout for titles\]](#), pagina [\[File structure\]](#), pagina [\[File structure\]](#).

File installati: `ly/titling-init.ly`.

### Formattazione predefinita delle intestazioni e dei piè di pagina

Le *intestazioni* e i *piè di pagina* sono linee di testo che appaiono in cima e in fondo alle pagine, distinte dal testo principale di un libro. Possono essere definite nelle seguenti variabili del blocco `\paper`:

- `oddHeaderMarkup`

- `evenHeaderMarkup`
- `oddFooterMarkup`
- `evenFooterMarkup`

Queste variabili di markup possono soltanto accedere ai campi testuali definiti nei blocchi `\header` del livello superiore (che vengono applicati a tutti i brani del libro) e sono definiti in `ly/titling-init.ly`. Impostazioni predefinite:

- i numeri di pagina sono collocati automaticamente in alto a sinistra (se la pagina è pari) o in alto a destra (se la pagina è dispari), a partire dalla seconda pagina.
- il campo `instrument` viene ripetuto al centro di ogni pagina, a partire dalla seconda pagina.
- il testo del `copyright` è centrato in fondo alla prima pagina.
- lo “slogan” (o firma) – `tagline` – è centrato in fondo all’ultima pagina o sotto il campo del `copyright` se c’è una sola pagina.

Il testo del piè di pagina predefinito per l’ultima pagina può essere modificato aggiungendo il campo `tagline` al blocco `\header` del livello superiore.

```
\book {
  \header {
    tagline = "... notazione musicale per tutti"
  }
  \score {
    \relative {
      c'4 d e f
    }
  }
}
```



... notazione musicale per tutti

Per toglierlo, impostare `tagline` su `##f`.

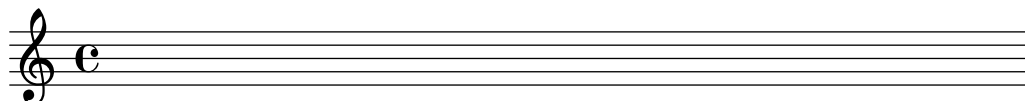
### 3.2.2 Titoli intestazioni e piè di pagina personalizzati

#### Titoli personalizzati

Si possono usare i normali comandi `\markup` per personalizzare qualsiasi intestazione, piè di pagina e titolo di un blocco `\header`.

```
\score {
  { s1 }
  \header {
    piece = \markup { \fontsize #4 \bold "PRAELUDIUM I" }
    opus = \markup { \italic "BWV 846" }
  }
}
```

}

**PRAELUDIUM I***BWV 846***Vedi anche**

Guida alla notazione: [\[Formatting text\]](#), pagina [\[undefined\]](#).

**Formattazione personalizzata dei titoli**

I comandi `\markup` nel blocco `\header` sono utili solo per la formattazione del testo, ma non consentono un controllo preciso sul posizionamento dei titoli. Per personalizzare il posizionamento dei campi testuali, cambiare una o entrambe le seguenti variabili `\paper`:

- `bookTitleMarkup`
- `scoreTitleMarkup`

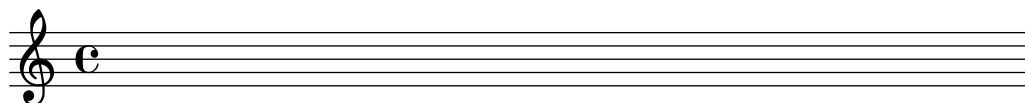
Il posizionamento dei titoli secondo i valori predefiniti di queste variabili `\markup` è mostrato negli esempi in [\[Default layout of bookpart and score titles\]](#), pagina [\[undefined\]](#).

Le impostazioni predefinite di `scoreTitleMarkup`, definite in `ly/titling-init.ly`, sono:

```
scoreTitleMarkup = \markup { \column {
  \on-the-fly \print-all-headers { \bookTitleMarkup \hspace #1 }
  \fill-line {
    \fromproperty #'header:piece
    \fromproperty #'header:opus
  }
}
```

Questo pone i campi testuali `piece` e `opus` alle estremità opposte della stessa riga:

```
\score {
  { s1 }
  \header {
    piece = "PRAELUDIUM I"
    opus = "BWV 846"
  }
}
```

**PRAELUDIUM I***BWV 846*

L'esempio seguente ridefinisce `scoreTitleMarkup` in modo che il campo testuale di `piece` sia centrato e in un tipo di carattere più grande e in grassetto.

```
\book {
  \paper {
    indent = 0\mm
```

```

scoreTitleMarkup = \markup {
  \fill-line {
    \null
    \fontsize #4 \bold \fromproperty #'header:piece
    \fromproperty #'header:opus
  }
}
\header { tagline = ##f }
\score {
  { s1 }
  \header {
    piece = "PRAELUDIUM I"
    opus = "BWV 846"
  }
}

```



I campi testuali che non sono normalmente attivi nei blocchi `\header` compresi in un blocco `\score` possono essere stampati nello spazio dedicato al titolo del brano aggiungendo `print-all-headers` nel blocco `\paper`. Lo svantaggio di questo metodo è che i campi testuali intesi esclusivamente per lo spazio del titolo della parte devono essere manualmente soppressi in ogni blocco `\score`. Vedi [\[Titles explained\]](#), pagina [\[undefined\]](#).

Per evitare ciò, è meglio mettere il campo testuale desiderato nella definizione di `scoreTitleMarkup`. Nell'esempio seguente il campo `composer` (solitamente associato a `bookTitleMarkup`) viene aggiunto a `scoreTitleMarkup`, facendo sì che ogni brano possa elencare un diverso compositore:

```

\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \fontsize #4 \bold \fromproperty #'header:piece
        \fromproperty #'header:composer
      }
    }
  }
}
\header { tagline = ##f }
\score {
  { s1 }
  \header {
    piece = "MENUET"
    composer = "Christian Petzold"
  }
}

```

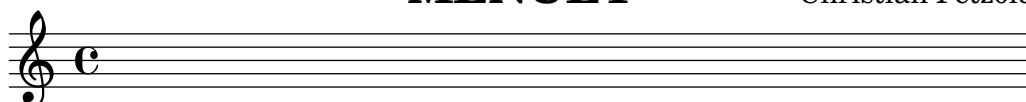
```

\score {
  { s1 }
  \header {
    piece = "RONDEAU"
    composer = "François Couperin"
  }
}

```

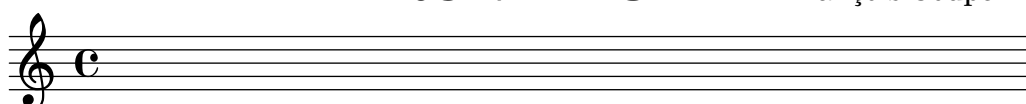
## MENUET

Christian Petzold



## RONDEAU

François Couperin



È anche possibile creare un campo testuale personalizzato e fare riferimento ad esso nella definizione di markup.

```

\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \override #`(direction . ,UP) {
          \dir-column {
            \center-align \fontsize #-1 \bold
            \fromproperty #'header:mycustomtext %% Campo definito dall'utente
            \center-align \fontsize #4 \bold
            \fromproperty #'header:piece
          }
        }
      }
      \fromproperty #'header:opus
    }
  }
}

\header { tagline = ##f }
\score {
  { s1 }
  \header {
    piece = "FUGA I"
    mycustomtext = "A 4 VOCI" %% Campo definito dall'utente
    opus = "BWV 846"
  }
}

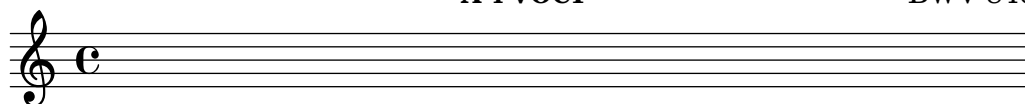
```



# FUGA I

A 4 VOCI

BWV 846



## Vedi anche

Guida alla notazione: [\[Titles explained\]](#), pagina [\[undefined\]](#).

## Formattazione personalizzata di intestazioni e piè di pagina

I comandi `\markup` nel blocco `\header` sono utili solo per la formattazione del testo, ma non consentono un controllo preciso sul posizionamento di intestazioni e piè di pagina. Per personalizzare il posizionamento dei campi testuali, usare una o entrambe le seguenti variabili `\paper`:

- `oddHeaderMarkup`
- `evenHeaderMarkup`
- `oddFooterMarkup`
- `evenFooterMarkup`

Il comando `\on-the-fly` – usato all'interno di un blocco `\markup` – permette di aggiungere del testo a intestazioni e piè di pagina definiti nel blocco `\paper`, solo se certe condizioni sono soddisfatte, tramite la seguente sintassi:

```
variabile = \markup {
  ...
  \on-the-fly \procedura testo
  ...
}
```

La *procedura* viene chiamata ogni volta che viene elaborato il comando `\markup` nel quale essa si trova. La *procedura* verifica una precisa condizione e interpreta (ovvero stampa) l'argomento *testo* se e solo se la condizione è vera.

Sono disponibili varie procedure pronte per verificare varie condizioni:

Nome della procedura	Condizione verificata
<code>print-page-number-check-first</code>	stampare questa pagina?
<code>create-page-number-stencil</code>	<code>print-page-numbers</code> è vero?
<code>print-all-headers</code>	<code>print-all-headers</code> è vero?
<code>first-page</code>	prima pagina del libro?
<code>(on-page nmbr)</code>	numero pagina = <code>nmbr</code> ?
<code>last-page</code>	ultima pagina del libro?
<code>not-first-page</code>	non la prima pagina del libro?
<code>part-first-page</code>	prima pagina della parte?
<code>part-last-page</code>	ultima pagina della parte?
<code>not-single-page</code>	pagine della parte > 1?

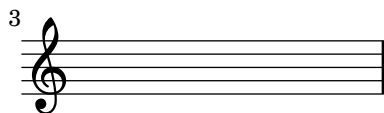
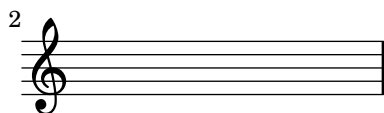
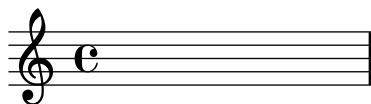
L'esempio seguente centra i numeri di pagina in fondo a ogni pagina. Innanzitutto vengono annullate le impostazioni predefinite per `oddHeaderMarkup` e `evenHeaderMarkup` definendo ciascuno di essi come un markup *null*. Poi `oddFooterMarkup` viene ridefinito col numero di pagina centrato. Infine a `evenFooterMarkup` viene assegnata la stessa formattazione di `\oddFooterMarkup`:

```
\book {
  \paper {
    print-page-number = ##t
```

```

print-first-page-number = ##t
oddHeaderMarkup = \markup \null
evenHeaderMarkup = \markup \null
oddFooterMarkup = \markup {
  \fill-line {
    \on-the-fly \print-page-number-check-first
    \fromproperty #'page:page-number-string
  }
}
evenFooterMarkup = \oddFooterMarkup
}
\score {
  \new Staff { s1 \break s1 \break s1 }
}
}

```



### 1

Varie condizioni `\on-the-fly` possono essere combinate insieme come se si utilizzasse l'operatore logico 'AND' e il testo apparirà solo se tutte le condizioni sono vere. Per esempio, queste due condizioni

```

\on-the-fly \first-page
\on-the-fly \last-page
{ \markup ... \fromproperty #'header: ... }

```

verificano se l'output è una pagina singola.

## Vedi anche

Guida alla notazione: [\[Titles explained\]](#), pagina [\[Default layout of bookpart and score titles\]](#), pagina [\[Default layout of bookpart and score titles\]](#).

File installati: `../ly/titling-init.ly`.

### 3.2.3 Creazione di metadati PDF

Oltre ad apparire nell'output, le variabili di `\header` vengono usate anche per impostare i metadati PDF (le informazioni mostrate dai lettori PDF come proprietà del file PDF). Per esempio, impostando la proprietà `title` del blocco `header` 'Sinfonia I' si assegnerà questo titolo anche al documento PDF.

```

\header{

```

```

    title = "Sinfonia I"
}

```

Se si desidera impostare il titolo dell'output su un valore e la proprietà titolo del PDF su un valore diverso, si può usare `pdftitle`, nel modo seguente:

```

\header{
  title = "Sinfonia I"
  pdftitle = "Sinfonia I di Beethoven"
}

```

Le variabili `title`, `subject`, `keywords`, `subtitle`, `composer`, `arranger`, `poet`, `author` e `copyright` impostano anche le proprietà del PDF e possono essere tutte prefissate con 'pdf' per impostare una proprietà del PDF su un valore diverso da quello dell'output.

La proprietà PDF `Creator` è automaticamente impostata su 'LilyPond' più la versione di LilyPond utilizzata, e `CreationDate` (data di creazione) e `ModDate` (data di modifica) sono entrambe impostate sulla data e ora correnti. `ModDate` può essere sovrascritta impostando nel blocco `header` la variabile `moddate` (o `pdfmoddate`) su una valida data per il PDF.

### 3.2.4 Creazione di note a piè di pagina

Le note a piè di pagina possono essere usate in situazioni diverse. In tutti i casi, un 'segno della nota a piè di pagina' viene inserito come riferimento vicino al testo o alla musica a cui si riferisce e il corrispondente 'testo della nota a piè di pagina' appare in fondo alla stessa pagina.

Le note a piè di pagina si creano diversamente a seconda che siano applicate a espressioni musicali o a del testo separato e fuori dalle espressioni musicali.

## Note a piè di pagina nelle espressioni musicali

### *Panoramica sulle note a piè di pagina attaccate alla musica*

Le note a piè di pagina nelle espressioni musicali appartengono a due categorie:

#### *Note a piè di pagina basate su un evento*

sono collegate a un preciso evento. Esempi di tali eventi sono note singole, articolazioni (come le indicazioni di diteggiatura, gli accenti, le dinamiche), e ciò che è successivo a un evento (come le legature di portamento e le travature manuali). La forma generale per le note a piè di pagina basate su un evento è la seguente:

```
[direzione] \footnote [segno] offset nota musica
```

#### *Note a piè di pagina basate sul tempo*

sono collegate a un preciso momento temporale in un contesto musicale. Alcuni comandi come `\time` e `\clef` non usano in realtà degli eventi per creare oggetti come le indicazioni di tempo e le chiavi. E nemmeno un accordo crea un suo evento: il suo gambo o coda sono creati alla fine di un'unità di tempo (attraverso un evento di una delle note al suo interno). Non è definito esattamente quale dei molteplici eventi nota di un accordo sarà giudicato la causa ultima di un gambo o di una coda. Dunque per legare una nota a questi, sono preferibili le note a piè di pagina basate sul tempo.

Una nota a piè di pagina basata sul tempo permette a tali oggetti della formattazione di avere delle note senza che sia necessario riferirsi a un evento. La forma generale per le note a piè di pagina basate sul tempo è la seguente:

```
\footnote [segno] offset nota [Contesto].NomeGrob
```

Gli elementi di entrambe le forme sono:

- direzione* se (e solo se) `\footnote` viene applicato a un post-evento o a una articolazione, deve essere preceduto da un indicatore di direzione (`-`, `_`, `^`) per poter collegare la *musica* (col segno della nota a piè di pagina) alla nota o pausa precedenti.
- segno* è del testo `-` racchiuso tra virgolette o in un blocco `\markup` – che specifica il segno della nota usata per contrassegnare sia il punto di riferimento che la nota stessa in fondo alla pagina. Può essere omesso (o sostituito con `\default`), nel qual caso sarà generato automaticamente un numero in sequenza. Tali sequenze numeriche ricominciano su ogni pagina contenente una nota.
- offset* è una coppia di numeri come `'#(2 . 1)'` che specificano la distanza orizzontale e verticale (X e Y offset), in unità di spazio rigo, dal bordo dell'oggetto in cui il segno deve essere posizionato. Valori positivi degli offset vengono presi dal bordo in alto a destra, valori negativi dal bordo in basso a sinistra e lo zero significa che il segno è centrato sul bordo.
- Contesto* è il contesto in cui è creato il grob a cui è collegata la nota. Può essere omesso se il grob si trova in un contesto più basso, per esempio un contesto **Voice**.
- NomeGrob* indica un tipo di grob a cui assegnare la nota (come `'Flag'`). Se viene specificato, la nota a piè di pagina non è collegata a un'espressione musicale in particolare, bensì a tutti i grob di quel tipo che si trovano in quel momento del tempo musicale.
- nota* è il testo – racchiuso tra virgolette o in un blocco `\markup` – che contiene il testo da usare per la nota a piè di pagina.
- musica* è l'evento musicale o il post-evento o articolazione a cui viene collegata la nota.

### ***Note a piè di pagina basate su un evento***

Una nota a piè di pagina può essere collegata a un oggetto della formattazione direttamente causato dall'evento corrispondente a *musica* con la sintassi:

```
\footnote [segno] offset nota musica
\book {
  \header { tagline = ##f }
  \relative c'' {
    \footnote #'(-1 . 3) "Una nota" a4
    a4
    \footnote #'(2 . 2) "Una pausa" r4
    a4
  }
}
```

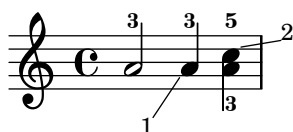



---

<sup>1</sup>Una nota  
<sup>2</sup>Una pausa

Contrassegnare un *intero* accordo con una nota a piè di pagina basata su un evento non è possibile: infatti un accordo, perfino uno che contenga una sola nota, non produce un vero evento specifico. Tuttavia possono essere contrassegnate singole note *dentro* l'accordo:

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    \footnote #'(2 . 3) "Non funziona" <a-3>2
    <\footnote #'(-2 . -3) "Funziona" a-3>4
    <a-3 \footnote #'(3 . 1/2) "Anche questo funziona" c-5>4
  }
}
```




---

<sup>1</sup>Funziona  
<sup>2</sup>Anche questo funziona

Se la nota deve essere attaccata a un post-evento o un'articolazione il comando `\footnote` deve essere preceduto da un indicatore di direzione, `-`, `_`, `^`, e seguito dal post-evento o dall'articolazione che si desidera commentare nell'argomento *musica*. In questa forma `\footnote` può essere considerato semplicemente una copia del suo ultimo argomento con un segno di nota a piè di pagina collegato a esso. La sintassi è:

*direzione* `\footnote` [*segno*] *offset* *nota* *musica*

```
\book {
  \header { tagline = ##f }
  \relative {
    a'4_\footnote #'(0 . -1) "Una legatura di portamento forzata in giù" (
    b8^_\footnote #'(1 . 0.5) "Una travatura manuale forzata in su" [
    b8 ]
    c4 )
    c-\footnote #'(1 . 1) "Tenuto" --
  }
}
```




---

<sup>1</sup>Una legatura di portamento forzata in giù  
<sup>2</sup>Una travatura manuale forzata in su  
<sup>3</sup>Tenuto

### *Note a piè di pagina basate sul tempo*

Se l'oggetto della formattazione a cui attaccare la nota è *indirettamente* causato da un evento – come un **Accidental** (alterazione) o **Stem** (gambo) causati da un evento **NoteHead** (testa di nota), è necessario specificare il **NomeGrob** dell'oggetto di formattazione al posto di *musica* dopo il testo della nota:

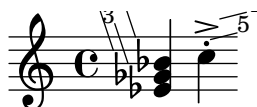
```
\book {
  \header { tagline = ##f }
  \relative c'' {
    \footnote #'(-1 . -3) "Un bemolle" Accidental
    aes4 c
    \footnote #'(-1 . 0.5) "Un altro bemolle" Accidental
    ees
    \footnote #'(1 . -2) "Un gambo" Stem
    aes
  }
}
```



- 
- <sup>1</sup>Un bemolle
  - <sup>2</sup>Un altro bemolle
  - <sup>3</sup>Un gambo

Tuttavia nota che, quando si specifica un **NomeGrob**, una nota a piè di pagina sarà attaccata a tutti i grob di quel tipo che si trovano in quel momento musicale:

```
\book {
  \header { tagline = ##f }
  \relative c' {
    \footnote #'(-1 . 3) "Un bemolle" Accidental
    <ees ges bes>4
    \footnote #'(2 . 0.5) "Articolazione" Script
    c'->-.
  }
}
```



- 
- <sup>1</sup>Un bemolle
  - <sup>2</sup>Un bemolle
  - <sup>3</sup>Un bemolle
  - <sup>4</sup>Articolazione
  - <sup>5</sup>Articolazione

È possibile assegnare a una nota di un accordo una singola nota a piè di pagina (basata su un evento). ‘NoteHead’ è l’unico grob causato direttamente dalla nota di un accordo, dunque una nota a piè di pagina basata su un evento è adatta *soltanto* ad aggiungere una nota a piè di pagina al ‘NoteHead’ all’interno di un accordo. Tutti gli altri grob delle note di un accordo sono causati indirettamente. Il comando `\footnote` non ha una sintassi per specificare *sia* un particolare tipo di grob *sia* un particolare evento a cui collegare la nota. Tuttavia si può usare un comando `\footnote` basato sul tempo per specificare il tipo di grob e poi precedere tale comando con `\single` perché venga applicato soltanto all’evento che segue:

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    < \footnote #'(1 . -2) "An A" a
      \single \footnote #'(-1 . -1) "Un diesis" Accidental
      cis
      \single \footnote #'(0.5 . 0.5) "Un bemolle" Accidental
      ees fis
    >2
  }
}
```



<sup>1</sup>Un bemolle  
<sup>2</sup>Un diesis  
<sup>3</sup>An A

**Nota:** Quando le note a piè di pagina sono collegate a diversi elementi musicali nello stesso momento musicale, come nell’esempio precedente, le note sono numerate dall’elemento più alto a quello più in basso come questi appaiono nell’output e non nell’ordine in cui sono inseriti nell’input.

Gli oggetti della formattazione come le chiavi e i cambi di armatura di chiave sono causati principalmente da proprietà modificate piuttosto che da veri eventi. Per questo motivo le note su tali oggetti devono essere basate sul loro tempo musicale. Le note a piè di pagina basate sul tempo sono da preferire anche quando si vogliono contrassegnare elementi come i gambi e le travature in un *accordo*: sebbene tali elementi dell’accordo siano nominalmente assegnati a *un* evento all’interno dell’accordo, affidarsi a una scelta particolare sarebbe imprudente.

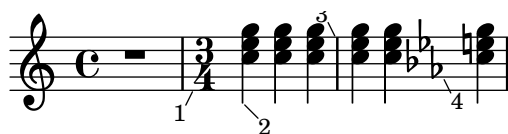
L’oggetto della formattazione in questione deve essere sempre specificato esplicitamente nelle note a piè di pagina basate sul tempo, e il contesto appropriato deve essere indicato se il grob viene creato in un contesto diverso da quello più basso.

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    r1 |
    \footnote #'(-0.5 . -1) "Cambio di tempo" Staff.TimeSignature
```

```

\time 3/4
\footnote #'(1 . -1) "Gambo dell'accordo" Stem
<c e g>4 q q
\footnote #'(-0.5 . 1) "Stanghetta" Staff.BarLine
q q
\footnote #'(0.5 . -1) "Cambio di armatura" Staff.KeySignature
\key c \minor
q
}
}

```



- 
- <sup>1</sup>Cambio di tempo  
<sup>2</sup>Gambo dell'accordo  
<sup>3</sup>Stanghetta  
<sup>4</sup>Cambio di armatura

Si possono usare segni personalizzati in alternativa a quelli numerici, e si può sopprimere la linea che collega l'oggetto commentato al segno:

```

\book {
  \header { tagline = ##f }
  \relative c' {
    \footnote "*" #'(0.5 . -2) \markup { \italic "*" La prima nota" } a'4
    b8
    \footnote \markup { \super "$" } #'(0.5 . 1)
    \markup { \super "$" \italic " La seconda nota" } e
    c4
    \once \override Score.FootnoteItem.annotation-line = ##f
    b-\footnote \markup \tiny "+" #'(0.1 . 0.1)
    \markup { \super "+" \italic " Editoriale" } \p
  }
}

```



- 
- \* *La prima nota*  
 \$ *La seconda nota*  
 + *Editoriale*

Altri esempi di segni personalizzati si trovano in [Footnotes in stand-alone text](#), pagina [Footnotes in stand-alone text](#).



## Note a piè di pagina nel testo separato

Vengono usate all'interno di blocchi `\markup` che si trovano fuori dalle espressioni musicali. Non hanno una linea che le unisce al loro punto di riferimento: i loro segni seguono semplicemente il testo citato. I segni possono essere inseriti automaticamente, nel qual caso sono numerici; altrimenti è possibile inserire manualmente dei segni personalizzati.

Le note a piè di pagina su testo separato vengono create in modo diverso a seconda che si scelgano segni automatici oppure segni personalizzati.

### *Note a piè di pagina nel testo separato con segni automatici*

La sintassi di una nota a piè di pagina nel testo separato con segni automatici è

```
\markup { ... \auto-footnote testo nota ... }
```

Gli elementi sono:

*testo*            è il testo – racchiuso tra virgolette doppie o in un blocco markup – da contrassegnare.

*nota*            è il testo della nota a piè di pagina.

Per esempio:

```
\book {
  \header { tagline = ##f }
  \markup {
    "Una semplice"
    \auto-footnote "canzone" \italic " Scritta da me"
    "è mostrata sotto. È una composizione"
    \auto-footnote "recente" \italic " Agosto 2012"
    "."
  }
  \relative {
    a'4 b8 e c4 d
  }
}
```

Una semplice canzone<sup>1</sup> è mostrata sotto. È una composizione recente<sup>2</sup>.




---

<sup>1</sup> *Scritta da me*

<sup>2</sup> *Agosto 2012*

### *Note a piè di pagina nel testo separato con segni personalizzati*

La sintassi di una nota a piè di pagina nel testo separato con segni personalizzati è

```
\markup { ... \footnote segno nota ... }
```

Gli elementi sono:

- segno* è una stringa di testo o un markup che indicano il segno da usare per contrassegnare il punto di riferimento. Tale segno *non* viene inserito automaticamente prima della nota stessa.
- nota* è una stringa di testo o un markup che indicano il testo della nota a piè di pagina, preceduto dal *segno*.

Qualsiasi carattere facile da scrivere, come \* o +, può essere usato come segno, come è spiegato in [Footnotes in music expressions](#), pagina [Footnotes in music expressions](#). Altrimenti, si possono usare gli alias ASCII (vedi [ASCII aliases](#), pagina [ASCII aliases](#)):

```
\book {
  \paper { #(include-special-characters) }
  \header { tagline = ##f }
  \markup {
    "Una semplice canzone"
    \footnote "*" \italic "*" Scritta da me"
    "è mostrata sotto. È una composizione recente"
    \footnote \super &dagger; \concat {
      \super &dagger; \italic " Agosto 2012"
    }
    "."
  }
  \relative {
    a'4 b8 e c4 d
  }
}
```

Una semplice canzone \* è mostrata sotto. È una composizione recente †.




---

\* *Scritta da me*

† *Agosto 2012*

È possibile usare anche i caratteri Unicode per indicare i segni (vedi [Unicode](#), pagina [Unicode](#)):

```
\book {
  \header { tagline = ##f }
  \markup {
    "Una semplice canzone"
    \footnote \super \char"U+2027" \concat {
      \super \char"U+2027" \italic " Scritta da me"
    }
  }
}
```

```

"è mostrata sotto. È una composizione recente"
\footnote \super \char##x00b6 \concat {
  \super \char##x00b6 \italic " Agosto 2012"
}
"."
}
\relative {
  a'4 b8 e c4 d
}
}

```

Una semplice canzone § è mostrata sotto. È una composizione recente ¶.




---

§ *Scritta da me*  
 ¶ *Agosto 2012*

## Vedi anche

Manuale di apprendimento: Sezione “Oggetti e interfacce” in *Manuale di Apprendimento*.

Guida alla notazione: <undefined> [ASCII aliases], pagina <undefined>, <undefined> [Balloon help], pagina <undefined>, <undefined> [List of special characters], pagina <undefined>, <undefined> [Text marks], pagina <undefined>, <undefined> [Text scripts], pagina <undefined>, [Unicode], pagina 509.

Guida al funzionamento interno: Sezione “FootnoteEvent” in *Guida al Funzionamento Interno*, Sezione “FootnoteItem” in *Guida al Funzionamento Interno*, Sezione “FootnoteSpanner” in *Guida al Funzionamento Interno*, Sezione “Footnote\_engraver” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Varie note a piè di pagina sulla stessa pagina possono essere messe soltanto una dopo l'altra; non è possibile disporle sulla stessa riga.

Le note a piè di pagina non possono essere collegate a `MultiMeasureRests`, a travature automatiche o a testo vocale.

I segni delle note a piè di pagina potrebbero entrare in collisione con righe, oggetti `\markup`, altri segni e linee delle note a piè di pagina.

### 3.2.5 Riferimento ai numeri di pagina

È possibile contrassegnare un punto specifico di una partitura usando il comando `\label` (etichetta) nel livello superiore o all'interno della musica. Questa etichetta può quindi essere

citata all'interno di un blocco markup, per ottenere il numero di pagina in cui è stato inserito il punto contrassegnato, tramite il comando markup `\page-ref`.

```
\header { tagline = ##f }
\book {
  \label #'primoBranco
  \score {
    {
      c'1
      \pageBreak \mark A \label #'segnoA
      c'1
    }
  }
  \markup { Il primo brano inizia a pagina \page-ref #'primoBranco "0" "?" }
  \markup { Il segno A è a pagina \page-ref #'segnoA "0" "?" }
}
```



Il primo brano inizia a pagina 1  
Il segno A è a pagina 2

Il comando markup `\page-ref` prende tre argomenti:

1. l'etichetta – un simbolo scheme – per esempio `#'primoBranco`;
2. un testo markup da usare come misura di riferimento per stimare le dimensioni del testo;
3. un testo markup che verrà usato al posto del numero di pagina se l'etichetta non viene trovata.

Il motivo per cui è necessario una misura di riferimento è che, nel momento in cui vengono interpretati i testi (markup), le interruzioni di pagina non sono state decise, quindi i numeri di pagina non sono ancora noti. Per aggirare il problema, la vera interpretazione del testo viene rimandata a un momento successivo; tuttavia le dimensioni del testo devono essere conosciute prima, ecco perché serve una misura di riferimento per decidere tali dimensioni. Se il libro ha un numero di pagine compreso tra 10 e 99, tale misura sarà "00", ovvero un numero di due cifre.

## Comandi predefiniti

`\label`, `\page-ref`.

### 3.2.6 Indice

L'indice si include col comando `\markuplist \table-of-contents`. Gli elementi che devono apparire nell'indice si inseriscono col comando `\tocItem`, che può essere usato nel livello superiore o all'interno di un'espressione musicale.

```
\markuplist \table-of-contents
```

```
\pageBreak
```

```
\tocItem \markup "Primo brano"
\score {
  {
    c'4 % ...
    \tocItem \markup "Un punto preciso nel primo brano"
    d'4 % ...
  }
}
```

```
\tocItem \markup "Secondo brano"
\score {
  {
    e'4 % ...
  }
}
```

I testi markup usati per formattare l'indice sono definiti nel blocco `\paper`. Ce ne sono due predefiniti:

- `tocTitleMarkup`  
Usato per formattare il titolo dell'indice.  

```
tocTitleMarkup = \markup \huge \column {
  \fill-line { \null "Indice" \null }
  \null
}
```
- `tocItemMarkup`  
Usato per formattare gli elementi dell'indice.  

```
tocItemMarkup = \markup \fill-line {
  \fromproperty #'toc:text \fromproperty #'toc:page
}
```

Queste variabili possono essere entrambe modificate.

Ecco un esempio che mostra come cambiare il titolo dell'indice in francese:

```
\paper {
  tocTitleMarkup = \markup \huge \column {
    \fill-line { \null "Table des matières" \null }
    \hspace #1
  }
}
```

Ecco un esempio che mostra come cambiare il corpo dei caratteri nell'indice:

```
\paper {
  tocItemMarkup = \markup \large \fill-line {
    \fromproperty #'toc:text \fromproperty #'toc:page
  }
}
```

Nota il modo in cui sono citati il testo e il numero di pagina dell'elemento dell'indice nella definizione di `tocItemMarkup`.

Includendo il comando `\tocItemWithDotsMarkup` dentro `tocItemMarkup` lo spazio tra un elemento dell'indice e la sua pagina corrispondente sarà riempito con dei punti:

```
\header { tagline = ##f }
```

```

\paper {
  tocItemMarkup = \tocItemWithDotsMarkup
}

\book {
  \markuplist \table-of-contents
  \tocItem \markup { Allegro }
  \tocItem \markup { Largo }
  \markup \null
}

```

## Table of Contents

Allegro . . . . .	1
Largo . . . . .	1

Si possono anche definire comandi personalizzati con markup specifici per creare un indice più complesso. Nell'esempio seguente, viene definito un nuovo stile per inserire i nomi degli atti nell'indice di un'opera.

Una nuova variabile di markup (chiamata `tocActMarkup`) viene definita nel blocco `\paper`:

```

\paper {
  tocActMarkup = \markup \large \column {
    \hspace #1
    \fill-line { \null \italic \fromproperty #'toc:text \null }
    \hspace #1
  }
}

```

Viene quindi aggiunta una funzione musicale personalizzata (`tocAct`), che usa la nuova definizione di markup `tocActMarkup`:

```

tocAct =
#(define-music-function (text) (markup?)
  (add-toc-item! 'tocActMarkup text))

```

Un file di input di LilyPond, che usi queste definizioni personalizzate, potrebbe avere il seguente output:

## Table of Contents

### *Atto Primo*

Coro. Viva il nostro Alcide	1
Cesare. Presti omai l'Egizia terra	1

### *Atto Secondo*

Sinfonia	1
Cleopatra. V'adoro, pupille, saette d'Amore	1

Ecco un esempio del comando `\fill-with-pattern` usato nel contesto di un indice:

```

\paper {

```

```

tocItemMarkup = \markup { \fill-line {
  \override #'(line-width . 70)
  \fill-with-pattern #1.5 #CENTER . \fromproperty #'toc:text \fromproperty #'toc:page
}
}
}

```

## Vedi anche

File installati: `ly/toc-init.ly`.

## Comandi predefiniti

`\table-of-contents`, `\tocItem`.

## 3.3 Lavorare coi file di input

### 3.3.1 Inclusione di file LilyPond

Se un progetto è grande e complesso, conviene suddividerlo in file separati. Per citare un altro file si usa

```
\include "altrofile.ly"
```

La riga `\include "altrofile.ly"` equivale a incollare i contenuti di `altrofile.ly` nel file corrente nel punto in cui appare il comando `\include`. Per esempio, in un progetto complesso si possono scrivere file separati per ogni parte strumentale e creare un file per l’“intera partitura” che raccoglie i file dei singoli strumenti. Di norma il file incluso definisce una serie di variabili che poi diventano disponibili nel file della partitura completa. Le sezioni contrassegnate con delle etichette nei file inclusi possono essere usate in varie parti di una partitura, vedi `<undefined>` [Different editions from one source], pagina `<undefined>`.

I file nella directory di lavoro corrente possono essere citati indicando semplicemente il nome del file dopo il comando `\include`. I file in altre posizioni possono essere inclusi sia con un percorso assoluto che con un percorso relativo (ma come separatore delle directory occorre usare la barra, o slash, come in UNIX, piuttosto che la barra inversa, o backslash, come in DOS/Windows). Per esempio, se `cose.ly` si trova una directory prima della directory di lavoro corrente, usare

```
\include "../cose.ly"
```

oppure se le parti orchestrali incluse si trovano tutte in una sottodirectory chiamata `parti` all’interno della directory corrente, usare

```

\include "parti/VI.ly"
\include "parti/VII.ly"
... etc

```

I file che devono essere inclusi possono contenere essi stessi delle dichiarazioni `\include`. Per impostazione predefinita, queste dichiarazioni `\include` di secondo livello non sono interpretate finché non vengono portate nel file principale, dunque i nomi dei file che specificano devono essere tutti relativi alla directory del file principale, non alla directory del file incluso. Tuttavia tale comportamento può essere cambiato globalmente tramite l’opzione `-drelative-includes` da linea di comando (oppure aggiungendo `#(ly:set-option 'relative-includes #t)` in cima al file di input principale).

Quando `relative-includes` viene impostato su `#t`, il percorso di ogni comando `\include` sarà considerato relativo al file che contiene quel comando. Questo comportamento è raccomandato e diventerà il comportamento predefinito in una versione futura di lilypond.

È possibile includere sia file relativi alla directory principale sia file relativi a qualche altra directory impostando **relative-includes** su **#t** o **#f** nei punti giusti dei file. Per esempio, se è stata creata una libreria generale, **libA**, che usa altri file inclusi dal file di base di quella libreria, tali dichiarazioni **\include** dovranno essere precedute da **#{ly:set-option #relative-includes #t}** per poter essere interpretate correttamente quando riportate nel file **.ly** principale:

```
libA/
  libA.ly
  A1.ly
  A2.ly
  ...
```

quindi il file di base, **libA.ly**, conterrà

```
#{ly:set-option 'relative-includes #t}
\include "A1.ly"
\include "A2.ly"
...
% ritorna alle impostazioni predefinite
#{ly:set-option 'relative-includes #f}
```

Qualsiasi file **.ly** può quindi includere l'intera libreria semplicemente con

```
\include "~/libA/libA.ly"
```

Si possono ideare strutture di file più complesse facendo dei cambi nei punti giusti.

È possibile includere dei file anche da una directory che si trova in un percorso di ricerca specificato come opzione quando si lancia LilyPond da linea di comando. I file inclusi possono allora essere specificati usando soltanto il loro nome. Per esempio, per compilare con questo metodo il file **principale.ly** che include i file di una sottodirectory chiamata **parti**, entrare nella directory di **principale.ly** e eseguire questo comando

```
lilypond --include=parti principale.ly
```

e in **principale.ly** scrivere

```
\include "VI.ly"
\include "VII.ly"
... etc
```

I file che devono essere inclusi in molte partiture possono essere salvati nella directory di installazione di LilyPond **../ly**. La posizione di questa directory dipende dal tipo di installazione (vedi Sezione “Altre fonti di informazione” in *Manuale di Apprendimento*). Questi file possono quindi essere inclusi semplicemente nominandoli in una dichiarazione **\include**. Questo è il modo in cui sono inclusi i file che dipendono dalla lingua, come **english.ly**.

LilyPond include un certo numero di file quando si lancia il programma. Queste inclusioni non sono evidenti all'utente, ma i file possono essere identificati eseguendo **lilypond --verbose** dalla linea di comando. Così si vedrà un elenco di percorsi e file che LilyPond usa, insieme a tante altre informazioni. I più importanti di questi file sono trattati in Sezione “Altre fonti di informazione” in *Manuale di Apprendimento*. Tali file possono essere modificati, ma le modifiche saranno perse quando si installa una nuova versione di LilyPond.

Alcuni semplici esempi d'uso di **\include** si trovano in Sezione “Partiture e parti” in *Manuale di Apprendimento*.

## Vedi anche

Manuale di apprendimento: Sezione “Altre fonti di informazione” in *Manuale di Apprendimento*, Sezione “Partiture e parti” in *Manuale di Apprendimento*.



## Problemi noti e avvertimenti

Se a un file incluso viene assegnato un nome identico a uno dei file di installazione di LilyPond, quest'ultimo ha la precedenza.

### 3.3.2 Edizioni diverse da un unico sorgente

Esistono vari metodi per generare versioni diverse di una partitura dalla stessa sorgente di musica. Le variabili sono forse le più utili per combinare lunghe sezioni musicali e/o note. Le etichette (*tag*) sono più utili per selezionare una sezione da varie sezioni brevi alternative e possono essere usate anche per unire insieme dei brani in punti diversi.

Qualsiasi metodo venga usato, la separazione delle note dalla struttura della partitura permetterà di cambiare la struttura lasciando le note intatte.

## Uso delle variabili

Se le sezioni musicali sono definite in variabili, possono essere riutilizzate in varie parti della partitura, come è stato spiegato in Sezione “Organizzare i brani con le variabili” in *Manuale di Apprendimento*. Per esempio, una partitura vocale *a cappella* spesso comprende, ai fini delle prove, una riduzione per pianoforte delle parti, identiche alla musica vocale, dunque la musica deve essere inserita una volta sola. La musica definita in due variabili può essere combinata in un rigo, come è spiegato in [\(undefined\)](#) [Automatic part combining], pagina [\(undefined\)](#). Ecco un esempio:

```
sopranoMusic = \relative { a'4 b c b8( a) }
altoMusic = \relative { e'4 e e f }
tenorMusic = \relative { c'4 b e d8( c) }
bassMusic = \relative { a4 gis a d, }
allLyrics = \lyricmode {King of glo -- ry }
<<
  \new Staff = "Soprano" \sopranoMusic
  \new Lyrics \allLyrics
  \new Staff = "Alto" \altoMusic
  \new Lyrics \allLyrics
  \new Staff = "Tenore" {
    \clef "treble_8"
    \tenorMusic
  }
  \new Lyrics \allLyrics
  \new Staff = "Basso" {
    \clef "bass"
    \bassMusic
  }
  \new Lyrics \allLyrics
  \new PianoStaff <<
    \new Staff = "Destra" {
      \set Staff.printPartCombineTexts = ##f
      \partcombine
      \sopranoMusic
      \altoMusic
    }
    \new Staff = "Sinistra" {
      \set Staff.printPartCombineTexts = ##f
      \clef "bass"
      \partcombine
```

```

\tenorMusic
\bassMusic
}
>>
>>

```



Partiture separate che mostrino soltanto le parti vocali o soltanto quelle per pianoforte possono essere prodotte semplicemente cambiando le dichiarazioni della struttura, lasciando la notazione musicale intatta.

Nel caso di partiture lunghe, conviene mettere le definizioni delle variabili in file separati da includere, vedi [\[Including LilyPond files\]](#), pagina [\[undefined\]](#).

## Uso delle etichette

Il comando `\tag #'parteA` contrassegna un'espressione musicale col nome *parteA*. Le espressioni contrassegnate in questo modo possono essere incluse o rimosse in base al loro nome successivamente, usando `\keepWithTag #'nome` oppure `\removeWithTag #'nome`. Il risultato dell'applicazione di questi filtri alla musica etichettata è il seguente:

### Filtro

Musica etichettata preceduta da `\keepWithTag #'nome` o `\keepWithTag #'(nome1 nome2...)`

Musica etichettata preceduta da `\removeWithTag #'nome` o `\removeWithTag #'(nome1 nome2...)`

Musica etichettata non preceduta da `\keepWithTag` o `\removeWithTag`

### Risultato

Viene inclusa la musica non etichettata e quella etichettata con uno dei nomi specificati; la musica etichettata con un nome etichetta diverso viene esclusa.

Viene inclusa la musica non etichettata e quella non etichettata con uno dei nomi specificati; la musica etichettata con uno dei nomi specificati viene esclusa.

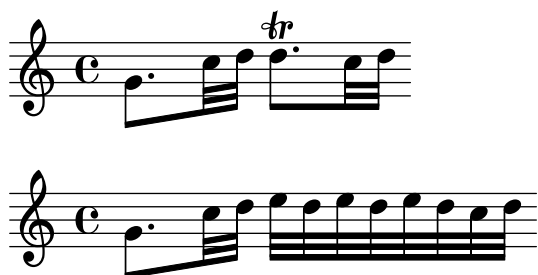
Viene inclusa tutta la musica etichettata e non etichettata.

Gli argomenti dei comandi `\tag`, `\keepWithTag` e `\removeWithTag` devono essere un simbolo o una lista di simboli (come `#'score` o `#'(violinoI violinoII)`), seguiti da un'espressione musicale. Se *e solo se* i simboli sono identificatori LilyPond validi (solo caratteri alfabetici, nessun numero, trattino o trattino basso) che non possono essere confusi con le note, si può omettere il `#'` e, come scorciatoia, una lista di simboli può usare il punto come separatore: quindi `\tag #'(violinoI violinoII)` può essere riscritto come `\tag violinoI.violinoII`. Lo stesso vale per `\keepWithTag` e `\removeWithTag`.

Nell'esempio seguente, vediamo due versioni di un brano musicale, una che mostra i trilli con la notazione abituale e l'altra con i trilli espansi esplicitamente:

```
musica = \relative {
  g'8. c32 d
  \tag #'trilli { d8.\trill }
  \tag #'espandi { \repeat unfold 3 { e32 d } }
  c32 d
}

\score {
  \keepWithTag #'trilli \musica
}
\score {
  \keepWithTag #'espandi \musica
}
```



Altrimenti, talvolta è più facile escludere sezioni musicali:

```
musica = \relative {
  g'8. c32 d
  \tag #'trilli { d8.\trill }
  \tag #'espandi { \repeat unfold 3 { e32 d } }
  c32 d
}

\score {
  \removeWithTag #'espandi
  \musica
}
\score {
  \removeWithTag #'trilli
  \musica
}
```





Il filtro delle etichette può essere applicato a articolazioni, testo, etc. scrivendo

```
-\tag #'tua-etichetta
```

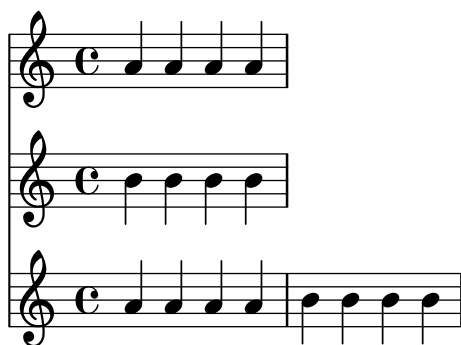
prima di un'articolazione. L'esempio seguente definisce una nota con un'indicazione di diteggiatura condizionale e una nota con un commento condizionale:

```
c1-\tag #'dito ^4
```

```
c1-\tag #'attenzione ^"Fai attenzione!"
```

Varie etichette possono essere associate a delle espressioni tramite molteplici `\tag`, o unendo molteplici etichette in una lista di simboli:

```
musica = \relative c'' {
  \tag #'a \tag #'entrambi { a4 a a a }
  \tag #'(b entrambi) { b4 b b b }
}
<<
\keepWithTag #'a \musica
\keepWithTag #'b \musica
\keepWithTag #'entrambi \musica
>>
```



Si possono applicare molteplici filtri `\removeWithTag` a una singola espressione musicale per togliere varie sezioni etichettate con nomi diversi. Altrimenti si può usare un solo `\removeWithTag` seguito da una lista di etichette.

```
musica = \relative c'' {
  \tag #'A { a4 a a a }
  \tag #'B { b4 b b b }
  \tag #'C { c4 c c c }
  \tag #'D { d4 d d d }
}
\new Voice {
  \removeWithTag #'B
  \removeWithTag #'C
  \musica
  \removeWithTag #'(B C)
  \musica
}
```



Due o più filtri `\keepWithTag` applicati a una singola espressione musicale toglieranno *tutte* le sezioni etichettate, perché il primo filtro toglie tutte le sezioni etichettate eccetto quella menzionata e il secondo filtro toglie anche quella sezione etichettata. Di solito si usa invece un singolo comando `\keepWithTag` con una lista di varie etichette: in questo modo verranno rimosse soltanto le sezioni etichettate non specificate nella lista.

Mentre `\keepWithTag` è comodo con *un* gruppo di alternative, la rimozione di musica contrassegnata con etichette *indipendenti* è problematico se si usano le etichette con diverse finalità. Per questa ragione, è possibile dichiarare ‘gruppi di etichette’ per raggruppare etichette correlate:

```
\tagGroup #'(violinoI violinoII viola cello)
```

dichiara che queste etichette appartengono a un gruppo.

```
\keepWithTag #'violinoI ...
```

filtrerà allora soltanto le etichette presenti nel gruppo cui appartiene `violinoI`: verrà tolto qualsiasi elemento che sia contrassegnato con una o più etichette di questo gruppo ma *non* con `violinoI`.

Per il comando `\keepWithTag`, sono visibili solo le etichette provenienti dai gruppi cui appartengono le etichette specificate dopo il comando.

Non è possibile assegnare la stessa etichetta a più di un gruppo.

Talvolta si ha necessità di combinare insieme della musica in un punto preciso di un’espressione musicale esistente. `\pushToTag` e `\appendToTag` permettono di aggiungere materiale prima o dopo gli elementi di un costrutto musicale esistente. Non tutti i costrutti musicali hanno elementi, ma nel caso di musica sequenziale e simultanea si può esserne sicuri:

```
test = { \tag #'qui { \tag #'qui <<c'>> } }
```

```
{
  \pushToTag #'qui c'
  \pushToTag #'qui e'
  \pushToTag #'qui g' \test
  \appendToTag #'qui c'
  \appendToTag #'qui e'
  \appendToTag #'qui g' \test
}
```



Entrambi i comandi prendono tre argomenti: un’etichetta, il materiale da combinare ad ogni occorrenza dell’etichetta e l’espressione contrassegnata.

## Vedi anche

Manuale di apprendimento: Sezione “Organizzare i brani con le variabili” in *Manuale di Apprendimento*.

Guida alla notazione: [\[Automatic part combining\]](#), pagina [\[undefined\]](#), [\[undefined\]](#) [\[Including LilyPond files\]](#), pagina [\[undefined\]](#).

## Problemi noti e avvertimenti

Se si usa `\relative` prima di un’espressione musicale ottenuta filtrando la musica con `\keepWithTag` o `\removeWithTag`, i rapporti di ottava potrebbero cambiare, perché verranno considerate solo le altezze rimaste nell’espressione filtrata. Per evitare questo rischio, usare `\relative` prima di `\keepWithTag` o `\removeWithTag`, in modo che `\relative` agisca su tutte le altezze prima del filtro.

## Impostazioni globali

È possibile includere impostazioni globali da un altro file:

```
lilypond -dinclude-settings=MIE_IMPOSTAZIONI.ly MIA_PARTITURA.ly
```

Gruppi di impostazioni come dimensioni del foglio e tipo di carattere possono essere salvati in file separati. Ciò permette di ottenere diverse edizioni dalla stessa partitura o di applicare delle impostazioni tipiche a molte partiture, semplicemente indicando il relativo file delle impostazioni.

Questa tecnica è la stessa usata per i fogli di stile, trattati in Sezione “Fogli di stile” in *Manuale di Apprendimento*.

## Vedi anche

Manuale di apprendimento: Sezione “Organizzare i brani con le variabili” in *Manuale di Apprendimento*, Sezione “Fogli di stile” in *Manuale di Apprendimento*.

Guida alla notazione: [\[Including LilyPond files\]](#), pagina [\[undefined\]](#).

### 3.3.3 Caratteri speciali

#### Codifica del testo

LilyPond usa il repertorio di caratteri definito dall’Unicode Consortium e dalla ISO/IEC 10646. Questo sistema di codifica definisce un nome e un numero univoci per gli insiemi di caratteri utilizzati in tutte le lingue moderne e anche in molte altre. Unicode può essere implementato in varie codifiche diverse. LilyPond usa la codifica UTF-8 (UTF sta per Unicode Transformation Format) che rappresenta tutti i comuni caratteri latini con un byte, e gli altri caratteri con un formato di lunghezza variabile fino a quattro byte.

L’aspetto grafico dei caratteri è determinato dai glifi definiti nei tipi di carattere disponibili - un tipo di carattere definisce la mappatura tra un sottoinsieme dei numeri Unicode e i glifi. LilyPond usa la libreria Pango per rappresentare e formattare i testi multilingua.

LilyPond non esegue alcuna conversione della codifica dell’input. Ciò significa che qualsiasi testo, sia esso un titolo, un testo vocale o un’istruzione musicale contenente caratteri non-ASCII, deve essere codificato in UTF-8. Il modo più semplice per inserire tale testo è usare un editor che sappia riconoscere la codifica Unicode e salvare il file con la codifica UTF-8. La maggior parte dei moderni editor supporta la codifica UTF-8, per esempio vim, Emacs, jEdit e Gedit. Tutti i sistemi MS Windows successivi a NT usano Unicode come codifica dei caratteri nativa, quindi perfino Notepad può modificare e salvare un file in formato UTF-8. Un’alternativa più efficiente per Windows è BabelPad.

Se un file di input LilyPond contenente un carattere non-ASCII non viene salvato in formato UTF-8, apparirà il seguente messaggio di errore:

```
FT_Get_Glyph_Name () error: invalid argument
```

Ecco un esempio che mostra del testo cirillico, ebraico e portoghese:



## Unicode

Per inserire un singolo carattere per il quale è noto il codice Unicode ma che non è disponibile nell’editor in uso, usare `\char ##xhhh` o `\char #dddd` dentro un blocco `\markup`, dove `hhh` è

il codice esadecimale del carattere richiesto e `dddd` è il valore decimale corrispondente. Gli zero iniziali possono essere omessi, ma di norma nella rappresentazione esadecimale si specificano tutti e quattro i caratteri. (Fare attenzione al fatto che la codifica UTF-8 del codice *non* deve essere usata dopo `\char`, perché le codifiche UTF-8 contengono bit extra che indicano il numero di ottetti.) Le tabelle dei codici Unicode e un indice dei nomi dei caratteri col proprio codice esadecimale sono disponibili sul sito dell'Unicode Consortium, <http://www.unicode.org/>.

Per esempio, `\char ##x03BE` e `\char #958` corrispondono entrambi al carattere Unicode U+03BE, che ha il nome Unicode “Greek Small Letter Xi”.

Qualsiasi codice Unicode può essere inserito in questo modo e se tutti i caratteri speciali sono inseriti in questo formato non è necessario salvare il file di input in formato UTF-8. Ovviamente, un tipo di carattere contenente tutti questi caratteri codificati deve essere installato e disponibile per LilyPond.

L'esempio seguente mostra valori esadecimali Unicode inseriti in quattro posti diversi: come numero di chiamata, come articolazione, nel testo vocale e come testo separato sotto il brano:

```
\score {
  \relative {
    c''1 \mark \markup { \char ##x03EE }
    c1_\markup { \tiny { \char ##x03B1 " a " \char ##x03C9 } }
  }
  \addlyrics { 0 \markup { \concat { Ph \char ##x0153 be! } } }
}
\markup { "Copyright 2008--2015" \char ##x00A9 }
```



Copyright 2008--2015 ©

Per inserire il segno del copyright nell'apposito campo usare:

```
\header {
  copyright = \markup { \char ##x00A9 "2008" }
}
```

## Alias ASCII

È possibile includere una lista di alias ASCII per i caratteri speciali:

```
\paper {
  #(include-special-characters)
}
```

```
\markup "&flqq; &ndash; &OE;uvre incomplète&hellip; &frqq;"
```

```
\score {
  \new Staff { \repeat unfold 9 a'4 }
  \addlyrics {
    Fun -- ziona an -- che nel~tes -- to: &ndash;_&OE;&hellip;
  }
}
```

```
\markup \column {
  "La sostituzione può essere disabilitata:"
  "&ndash; &OE; &hellip;"
  \override #'(replacement-alist . ()) "&ndash; &OE; &hellip;"
}
```

« – Œuvre incomplète... »



La sostituzione può essere disabilitata:

– Œ ...

&ndash; &OE; &hellip;

Si possono creare i propri alias, sia globalmente:

```
\paper {
  #(add-text-replacements!
    '(("100" . "hundred")
      ("dpi" . "dots per inch")))
}
\markup "A 100 dpi."
```

A hundred dots per inch.

che localmente:

```
\markup \replace #'(("100" . "hundred")
  ("dpi" . "dots per inch")) "A 100 dpi."
```

A hundred dots per inch.

## Vedi anche

Guida alla notazione: [\[List of special characters\]](#), pagina [\[undefined\]](#).

File installati: `ly/text-replacements.ly`.

## 3.4 Controllo dell'output

### 3.4.1 Estrarre frammenti musicali

È possibile creare output separati di uno o più frammenti di una partitura definendo i punti della musica da estrarre nel blocco `\layout` del file di input tramite la funzione `clip-regions`, e poi eseguendo LilyPond con l'opzione `-dclip-systems`;

```
\layout {
  clip-regions
  = #(list
    (cons
      (make-rhythmic-location 5 1 2)
      (make-rhythmic-location 7 3 4)))
```



```
}
```

Questo esempio estrarrà dal file di input un unico frammento che *inizia* dopo una minima nella quinta misura (5 1 2) e *termina* dopo la terza semiminima nella settima misura (7 3 4).

Si possono estrarre ulteriori frammenti aggiungendo altre coppie di `make-rhythmic-location` alla lista `clip-regions` del blocco `\layout`.

Ciascun frammento musicale verrà salvato in formato EPS, ma se necessario si possono creare anche altri formati come PDF o PNG. La musica estratta viene generata come se fosse letteralmente ‘tagliata’ dalla partitura a stampa originale; ciò significa che se un frammento supera una o più linee, verrà creato un file di output separato per ciascuna linea.

## Vedi anche

Guida alla notazione: `<undefined>` [The layout block], pagina `<undefined>`.

Utilizzo: Sezione “Uso da linea di comando” in *Uso del Programma*.

### 3.4.2 Saltare la musica già corretta

Quando si inserisce o si copia della musica, di solito è utile visualizzare e correggere soltanto la musica vicina alla fine (ovvero dove si stanno inserendo le note). Per velocizzare il processo di correzione, è possibile far sì che il compositore tipografico salti tutte le misure eccetto le ultime. Per farlo basta inserire per esempio

```
showLastLength = R1*5
\score { ... }
```

nel file sorgente. In questo modo verranno elaborate soltanto le ultime 5 misure (assumendo che il tempo sia 4/4) di ogni blocco `\score` nel file di input. Per i brani più lunghi, elaborare solo una piccola parte è spesso molto più veloce di elaborarli completamente. Se si lavora sull’inizio di un brano già scritto (per esempio per aggiungere una nuova parte), si userà invece la proprietà `showFirstLength`.

È possibile saltare parti di una partitura in un modo più preciso tramite la proprietà `Score.skipTypesetting`. Quando è impostata su vero, la composizione tipografica è disattivata.

Questa proprietà viene usata anche per controllare l’output da inviare al file MIDI. Attenzione: salta tutti gli eventi, inclusi i cambi di tempo e di strumento.

```
\relative c' {
  c1
  \set Score.skipTypesetting = ##t
  \tempo 4 = 80
  c4 c c c
  \set Score.skipTypesetting = ##f
  d4 d d d
}
```



Nella musica polifonica, `Score.skipTypesetting` agisce su tutte le voci e su tutti i righi, facendo risparmiare tempo ulteriormente.

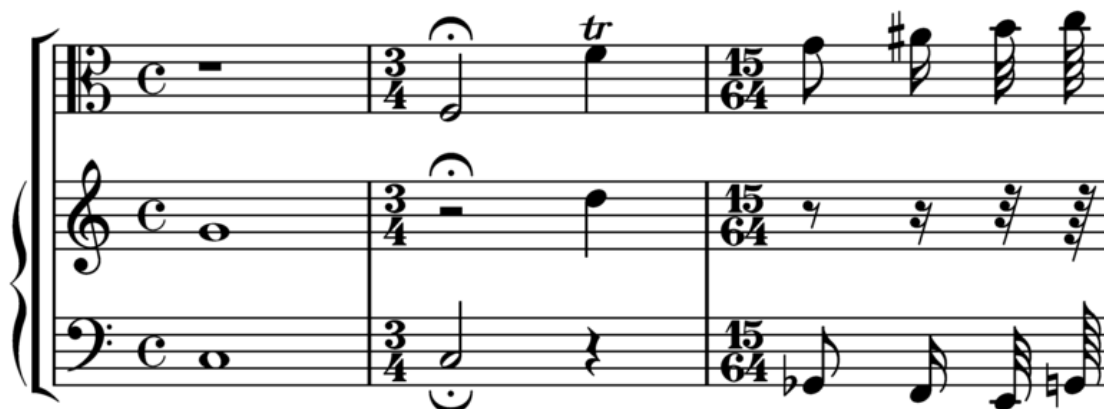
### 3.4.3 Formati di output alternativi

I formati di output predefiniti per la partitura stampata sono Portable Document Format (PDF) e PostScript (PS). Sono disponibili anche i formati di output Scalable Vector Graphics (SVG), Encapsulated PostScript (EPS) e Portable Network Graphics (PNG) attraverso le opzioni della linea di comando, vedi Sezione “Opzioni di base della linea di comando per LilyPond” in *Uso del Programma*.

### 3.4.4 Cambiare il tipo di carattere della notazione

Gonville è un’alternativa al tipo di carattere Feta usato da LilyPond e può essere scaricato da: <http://www.chiark.greenend.org.uk/~sgtatham/gonville/> (<http://www.chiark.greenend.org.uk/~sgtatham/gonville/> )

Ecco alcune battute di musica che usa Gonville:



E alcune battute di musica che usa Feta (il tipo di carattere predefinito di LilyPond):



### Istruzioni di installazione per MacOS

Scaricare e estrarre il file zip. Copiare la directory `lilyfonts` in `SHARE_DIR/lilypond/current`; maggiori informazioni si trovano in Sezione “Altre fonti di informazione” in *Manuale di Apprendimento*. Rinominare la directory esistente `fonts` come `fonts_orig` e la directory `lilyfonts` come `fonts`. Per tornare a Feta, fare il contrario.

### Vedi anche

Manuale di apprendimento: Sezione “Altre fonti di informazione” in *Manuale di Apprendimento*.

## Problemi noti e avvertimenti

Gonville non può essere usato per scrivere notazione in ‘musica antica’ ed è probabile che i nuovi glifi introdotti in rilasci successivi di LilyPond non esistano in Gonville. Fare riferimento al sito web dell’autore per maggiori informazioni su queste e altre problematiche, inclusa la licenza di Gonville.

## 3.5 Creazione dell’output MIDI

LilyPond è in grado di produrre dei file conformi allo standard MIDI (Musical Instrument Digital Interface) e dunque permette di fare un controllo uditivo dell’output musicale (con l’aiuto di un’applicazione o un dispositivo che comprendano il MIDI). L’ascolto dell’output MIDI può aiutare anche a individuare errori, come note inserite in modo erraneo o note senza alterazioni e così via.

I file MIDI non contengono audio (come i file AAC, MP3 o Vorbis), bensì richiedono un software esterno che produca il suono a partire da essi.

### 3.5.1 Notazione supportata nel MIDI

I seguenti elementi della notazione musicale saranno resi nell’output MIDI usando le capacità predefinite di LilyPond:

- Respiri
- Accordi inseriti come nomi degli accordi
- Crescendi e decrescendi su varie note. Il volume viene modificato in modo proporzionato fra i due estremi
- Segni di dinamica da **ppppp** a **fffff**, inclusi **mp**, **mf** e **sf**
- Microtoni ma *non* gli accordi microtonali. È necessario anche un lettore MIDI che supporti la “piegatura” (*bending*) delle altezze.
- Testo vocale
- Altezze
- Ritmi inseriti come durate, inclusi i gruppi irregolari
- Articolazioni ‘semplici’: staccato, staccatissimo, accento, marcato e portato
- Cambi di tempo tramite la funzione `\tempo`
- Legature di valore
- Tremoli che *non* sono inseriti con un valore ‘:[numero]’

Alcuni effetti, come panning, bilanciamento, espressione, riverbero e chorus, si gestiscono impostando le proprietà di contesto, vedi `<undefined>` [Context properties for MIDI effects], pagina `<undefined>`.

Se si usa lo script `articulate`, anche i seguenti elementi della notazione musicale saranno presenti nell’output MIDI:

- Appoggiature. Servono a rubare metà del valore della nota che le segue (senza considerare i punti). Per esempio:  
`\appoggiatura c8 d2.`  
 Il c avrà il valore di una semiminima.
- Ornamenti (mordenti, trilli, gruppetti, etc.)
- Rallentando, accelerando, ritardando e a tempo
- Legature di portamento, incluse le legature di frase
- Tenuto

Vedi `<undefined>` [Enhancing MIDI output], pagina `<undefined>`.

### 3.5.2 Notazione non supportata nel MIDI

I seguenti elementi della notazione musicale non possono essere trasferiti nel file MIDI:

- Articolazioni diverse da staccato, staccatissimo, accento, marcato e portato
- Crescendi e decrescendi su una *singola* nota
- Corona
- Basso numerato (continuo)
- Glissandi
- Portamenti indeterminati verso il basso e verso l'alto
- Accordi microtonali
- Ritmi inseriti come note, per esempio lo swing
- Cambi di tempo senza usare `\tempo` (per esempio, inseriti come note)
- Tremoli che *sono* inseriti con un valore `':[numero]'`

### 3.5.3 Il blocco MIDI

Per creare un file MIDI da un file di input LilyPond, inserire un blocco `\midi`, che può essere vuoto, all'interno di un blocco `\score`;

```
\score {
  ... musica ...
  \layout { }
  \midi { }
}
```

**Nota:** Un blocco `\score` che, oltre alla musica, contenga soltanto un blocco `\midi` (ovvero *sia* *privo* del blocco `\layout`), produrrà come output soltanto file MIDI e nessun file visuale.

L'estensione predefinita del file di output (`.midi`) può essere modificata tramite l'opzione `-dmidi-extension` del comando `lilypond`:

```
lilypond -dmidi-extension=mid MyFile.ly
```

Altrimenti, aggiungere la seguente espressione Scheme prima dell'inizio dei blocchi `\book`, `\bookpart` o `\score`. Vedi [\[File structure\]](#), pagina [\[File structure\]](#).

```
#(ly:set-option 'midi-extension "mid")
```

### Vedi anche

Guida alla notazione: [\[File structure\]](#), pagina [\[File structure\]](#).

File installati: `scm/midi.scm`.

### Problemi noti e avvertimenti

Sono disponibili quindici canali MIDI e un canale ulteriore (`#10`) per le percussioni. I righi sono assegnati ai canali in sequenza: in una partitura di più di quindici righi, i righi extra condivideranno (senza sovrascriverlo) lo stesso canale MIDI. Ciò potrebbe essere un problema se i righi in questione sono impostati con proprietà MIDI in conflitto e basate sul canale, come ad esempio diversi strumenti MIDI.

### 3.5.4 Gestione delle dinamiche nel MIDI

È possibile regolare il volume MIDI complessivo, il volume relativo dei segni di dinamica e il volume relativo dei diversi strumenti.

Le dinamiche vengono trasferite automaticamente sui livelli di volume nella gamma di volume MIDI disponibile, mentre crescendi e decrescendi variano il volume gradualmente tra i loro due

estremi. È possibile regolare il volume relativo delle dinamiche e i livelli del volume complessivo dei diversi strumenti.

## Dinamiche nel MIDI

Solo i segni di dinamica compresi tra **ppppp** e **fffff**, inclusi **mp**, **mf** e **sf** hanno dei valori assegnati. Questo valore viene poi applicato al valore della gamma del volume MIDI complessivo per ottenere il volume finale incluso nell'output MIDI per quella particolare dinamica. Le frazioni predefinite vanno da 0.25 per **ppppp** a 0.95 per **fffff**. L'insieme completo di dinamiche e delle loro frazioni associate si trova in `scm/midi.scm`.

## Frammenti di codice selezionati

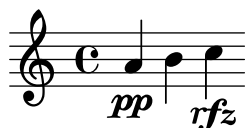
*Creare dinamiche personalizzate nell'output MIDI*

L'esempio seguente mostra come creare un segno di dinamica, non incluso nell'elenco predefinito, e assegnargli un valore specifico così che possa essere usato per cambiare l'output MIDI.

Al segno di dinamica `\rfz` viene assegnato il valore 0.9.

```
#(define (myDynamics dynamic)
  (if (equal? dynamic "rfz")
      0.9
      (default-dynamic-absolute-volume dynamic)))

\score {
  \new Staff {
    \set Staff.midiInstrument = #"cello"
    \set Score.dynamicAbsoluteVolumeFunction = #myDynamics
    \new Voice {
      \relative {
        a'4\pp b c-\rfz
      }
    }
  }
  \layout {}
  \midi {}
}
```



File installati: `ly/script-init.ly` `scm/midi.scm`.

Frammenti: Sezione “MIDI” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Dynamic\_performer” in *Guida al Funzionamento Interno*.

## Impostazione del volume MIDI

I valori minimo e massimo del volume complessivo delle dinamiche MIDI sono regolati dalle proprietà `midiMinimumVolume` e `midiMaximumVolume` nel livello `Score`. Tali proprietà hanno effetto soltanto all'inizio di una voce e sui segni di dinamica. La frazione corrispondente a ciascun segno di dinamica viene modificata con la seguente formula:

$$\text{midiMinimumVolume} + (\text{midiMaximumVolume} - \text{midiMinimumVolume}) * \text{frazione}$$

Nell'esempio seguente la gamma dinamica del volume MIDI complessivo è limitata alla gamma 0.2 - 0.5.

```
\score {
  <<
    \new Staff {
      \set Staff.midiInstrument = #"flauto"
      ... musica ...
    }
    \new Staff {
      \set Staff.midiInstrument = #"clarinetto"
      ... musica ...
    }
  >>
  \midi {
    \context {
      \Score
      midiMinimumVolume = #0.2
      midiMaximumVolume = #0.5
    }
  }
}
```

Una semplice equalizzazione degli strumenti MIDI si può ottenere impostando le proprietà `midiMinimumVolume` e `midiMaximumVolume` nel contesto `Staff`.

```
\score {
  \new Staff {
    \set Staff.midiInstrument = #"flauto"
    \set Staff.midiMinimumVolume = #0.7
    \set Staff.midiMaximumVolume = #0.9
    ... musica ...
  }
  \midi { }
}
```

In caso di partiture con molti righi e molti strumenti MIDI, i volumi relativi di ogni strumento possono essere impostati individualmente;

```
\score {
  <<
    \new Staff {
      \set Staff.midiInstrument = #"flauto"
      \set Staff.midiMinimumVolume = #0.7
      \set Staff.midiMaximumVolume = #0.9
      ... musica ...
    }
    \new Staff {
      \set Staff.midiInstrument = #"clarinetto"
      \set Staff.midiMinimumVolume = #0.3
      \set Staff.midiMaximumVolume = #0.6
      ... musica ...
    }
  >>
  \midi { }
}
```

In questo esempio il volume del clarinetto è diminuito in modo proporzionale al volume del flauto.

Se queste proprietà del volume non sono impostate, LilyPond applica comunque “un po” di equalizzazione a certi strumenti. Vedi `scm/midi.scm`.

File installati: `scm/midi.scm`.

## Vedi anche

Guida alla notazione: `<undefined>` [Score layout], pagina `<undefined>`.

Guida al funzionamento interno: Sezione “Dynamic-performer” in *Guida al Funzionamento Interno*.

## Frammenti di codice selezionati

*Modificare l'equalizzazione predefinita degli strumenti MIDI*

L'equalizzatore predefinito degli strumenti MIDI può essere modificato impostando la proprietà `instrumentEqualizer` nel contesto `Score` come una procedura Scheme definita dall'utente che usi il nome dello strumento MIDI come argomento insieme a una coppia di frazioni indicanti i volumi minimi e massimi da applicare a quello specifico strumento.

L'esempio seguente imposta i volumi massimo e minimo per il flauto e per il clarinetto.

```
#(define my-instrument-equalizer-alist '())

#(set! my-instrument-equalizer-alist
  (append
    '(
      ("flute" . (0.7 . 0.9))
      ("clarinet" . (0.3 . 0.6)))
    my-instrument-equalizer-alist))

#(define (my-instrument-equalizer s)
  (let ((entry (assoc s my-instrument-equalizer-alist)))
    (if entry
      (cdr entry))))

\score {
  <<
    \new Staff {
      \key g \major
      \time 2/2
      \set Score.instrumentEqualizer = #my-instrument-equalizer
      \set Staff.midiInstrument = #"flute"
      \new Voice \relative {
        r2 g''\mp g fis~
        4 g8 fis e2~
        4 d8 cis d2
      }
    }
  \new Staff {
    \key g \major
    \set Staff.midiInstrument = #"clarinet"
    \new Voice \relative {
      b'1\p a2. b8 a
    }
  }
}
```

```

        g2. fis8 e
        fis2 r
    }
}
>>
\layout { }
\midi { }
}

```



## Problemi noti e avvertimenti

Le modifiche al volume MIDI si verificano soltanto all'inizio di una nota, quindi i crescendo e i decrescendo non possono cambiare il volume di una singola nota.

## Impostazione delle proprietà del blocco MIDI

Il blocco `\midi` può contenere modifiche del contesto, nuove definizioni del contesto o codice che imposti i valori di certe proprietà.

```

\score {
  ... musica ...
  \midi {
    \tempo 4 = 72
  }
}

```

In questo esempio il tempo è impostato su 72 battiti da un quarto per minuto. Il segno di tempo nel blocco `\midi` non appare nella partitura, mentre qualsiasi altra indicazione di `\tempo` specificata nel blocco `\score` sarà trasferita anche nell'output MIDI.

Se all'interno di un blocco `\midi`, il comando `\tempo` imposta le proprietà durante l'interpretazione della musica e nel contesto delle definizioni di output; dunque viene interpretato *come se* fosse una modifica di contesto.

Le definizioni di contesto seguono la stessa sintassi di quelle di un blocco `\layout`;

```

\score {
  ... musica ...
  \midi {
    \context {
      \Voice
      \remove "Dynamic_performer"
    }
  }
}

```

Questo esempio toglie l'effetto delle dinamiche dall'output MIDI. Nota bene: i moduli di traduzione di LilyPond usati per l'audio si chiamano 'performer'.



## Vedi anche

Manuale di apprendimento: Sezione “Altre fonti di informazione” in *Manuale di Apprendimento*.

Guida alla notazione: [\[Expressive marks\]](#), pagina [\[Score layout\]](#), pagina [\[Score layout\]](#).

File installati: `ly/performer-init.ly`.

Frammenti: Sezione “MIDI” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Dynamic\_performer” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Alcuni lettori MIDI non sempre gestiscono correttamente i cambi di tempo.

I cambi di `midiInstrument`, e di altre opzioni MIDI, all'*inizio* di un rigo potrebbero apparire due volte nell'output MIDI.

### 3.5.5 Uso degli strumenti MIDI

Gli strumenti MIDI si impostano tramite la proprietà `midiInstrument` del contesto `Staff`.

```
\score {
  \new Staff {
    \set Staff.midiInstrument = #"glockenspiel"
    ... musica ...
  }
  \midi { }
}

o

\score {
  \new Staff \with {midiInstrument = #"cello"} {
    ... musica ...
  }
  \midi { }
}
```

Se il nome dello strumento non corrisponde a nessuno degli strumenti elencati nella sezione ‘Strumenti MIDI’, verrà usato lo strumento **acoustic grand**. Vedi [\[MIDI instruments\]](#), pagina [\[MIDI instruments\]](#).

## Vedi anche

Manuale di apprendimento: Sezione “Altre fonti di informazione” in *Manuale di Apprendimento*.

Guida alla notazione: [\[MIDI instruments\]](#), pagina [\[MIDI instruments\]](#), pagina [\[Score layout\]](#).

File installati: `scm/midi.scm`.

## Problemi noti e avvertimenti

Gli strumenti percussivi che sono scritti in un contesto `DrumStaff` verranno inviati, correttamente, al canale MIDI 10, ma alcuni strumenti percussivi con un tono musicale, come xilofono, marimba, vibrafono o timpano, sono trattati come strumenti “normali”, quindi la musica di tali strumenti deve essere inserita in un contesto `Staff` (non `DrumStaff`) per ottenere l'output MIDI corretto. L'elenco completo delle voci del set di percussioni del canale 10 `channel 10`

`drum-kits` si trova in `scm/midi.scm`. Vedi Sezione “Altre fonti di informazione” in *Manuale di Apprendimento*.

### 3.5.6 Uso delle ripetizioni nel MIDI

Le ripetizioni possono essere applicate all’output MIDI tramite il comando `\unfoldRepeats`.

```
\score {
  \unfoldRepeats {
    \repeat tremolo 8 { c'32 e' }
    \repeat percent 2 { c''8 d'' }
    \repeat volta 2 { c'4 d' e' f' }
    \alternative {
      { g' a' a' g' }
      { f' e' d' c' }
    }
  }
  \midi { }
}
```

Per limitare l’effetto di `\unfoldRepeats` all’output MIDI soltanto, senza modificare la partitura stampata, occorre creare *due* blocchi `\score`; uno per il MIDI (con ripetizioni ricopiate) e uno per la partitura scritta (con ripetizioni con volta, tremolo e percentuale):

```
\score {
  ... musica ...
  \layout { }
}
\score {
  \unfoldRepeats {
    ... musica ...
  }
  \midi { }
}
```

Quando si usa più di una voce, ciascuna voce deve avere tutte le ripetizioni ricopiate per generare un output MIDI corretto.

### Vedi anche

Guida alla notazione: `<undefined>` [Repeats], pagina `<undefined>`.

### 3.5.7 Mappatura dei canali MIDI

Quando genera un file MIDI da una partitura, LilyPond assegna automaticamente ogni nota a un canale MIDI, in cui essa dovrà essere suonata quando inviata a un dispositivo MIDI. Un canale MIDI mette a disposizione un certo numero di controlli per selezionare, per esempio, lo strumento da usare per riprodurre le note in quel canale o per chiedere al dispositivo MIDI di applicare vari effetti al suono prodotto nel canale. In ogni momento, ogni controllo di un canale MIDI può avere un solo valore assegnato (che tuttavia può essere modificato, per esempio, per passare a un altro strumento nel mezzo di un brano).

Lo standard MIDI supporta soltanto 16 canali per dispositivo MIDI. Questo limite al numero di canali limita anche il numero dei diversi strumenti che possono essere eseguiti in contemporanea.

LilyPond crea tracce MIDI separate per ogni rigo (o strumento o voce separati, a seconda del valore di `Score.midiChannelMapping`) e anche per ogni contesto lyrics. Non c’è limite al numero di tracce.

Per aggirare il limite del numero di canali MIDI, LilyPond supporta varie modalità per l'assegnazione dei canali MIDI, scelti attraverso la proprietà di contesto `Score.midiChannelMapping`. In ogni caso, se è necessario un numero di canali MIDI superiore al limite, i numeri canale assegnati ripartono da 0, causando possibili assegnazioni erranee degli strumenti di alcune note. Questa proprietà di contesto può essere impostata su uno dei seguenti valori:

#### `'staff`

Assegna un canale MIDI separato a ogni rigo della partitura (comportamento predefinito). Tutte le note in tutte le voci di ogni rigo condivideranno lo stesso canale MIDI del rigo che le racchiude, e si trovano tutte nella stessa traccia MIDI.

Il limite dei 16 canali è applicato al numero totale dei contesti staff (rigo) e lyrics (testo vocale), anche se il testo vocale nel MIDI non occupa un canale MIDI.

#### `'instrument`

Assegna un canale MIDI separato a ogni strumento MIDI diverso specificato nella partitura. Ciò significa che tutte le note suonate con lo stesso strumento MIDI condivideranno lo stesso canale (e traccia) MIDI, anche se le note appartengono a voci o righe diversi.

In questo caso i contesti del testo vocale (lyrics) non contano nel calcolo del limite dei 16 canali MIDI (perché non saranno assegnati a uno strumento MIDI), dunque questa impostazione consente una migliore assegnazione dei canali MIDI quando il numero di righe e contesti lyrics in una partitura è superiore a 16.

#### `'voice`

Assegna un canale MIDI separato a ogni voce che abbia un nome unico tra le voci del rigo che le racchiude. Alle voci in righe diversi vengono sempre assegnati canali MIDI separati, ma due voci di uno stesso rigo condivideranno lo stesso canale MIDI se hanno lo stesso nome. `midiInstrument` e i vari controlli MIDI per gli effetti, essendo proprietà del contesto del rigo (staff), non possono essere impostati separatamente per ogni voce. La prima voce verrà suonata con lo strumento e gli effetti specificati per il rigo, mentre alle voci con un nome diverso da quello della prima saranno assegnati lo strumento e gli effetti predefiniti.

Nota bene: è possibile assegnare diversi strumenti e/o effetti a varie voci dello stesso rigo spostando `Staff_performer` dal contesto `Staff` al contesto `Voice` e lasciando `midiChannelMapping` sul valore predefinito `'staff` oppure impostandolo su `'instrument`; vedi il frammento in basso.

Per esempio, la mappatura predefinita dei canali MIDI di una partitura può essere modificata per usare l'impostazione `'instrument`:

```
\score {
  ...musica...
  \midi {
    \context {
      \Score
      midiChannelMapping = #'instrument
    }
  }
}
```

## Frammenti di codice selezionati

*Impostare l'output MIDI su un canale per voce*

Nella creazione del file di output MIDI, il comportamento predefinito prevede che ogni rigo sia assegnato a un canale MIDI, con tutte le voci del rigo amalgamate in un canale. Ciò diminuisce il rischio di esaurire i canali MIDI disponibili, dato che ce ne sono solo 16 per traccia.

Tuttavia, spostando `Staff_performer` nel contesto `Voice`, ogni voce in un rigo può avere il proprio canale MIDI, come è illustrato nell'esempio seguente: sebbene le voci siano sullo stesso rigo, vengono creati due canali MIDI, ciascuno con un diverso strumento MIDI (`midiInstrument`).

```
\score {
  \new Staff <<
    \new Voice \relative c''' {
      \set midiInstrument = #"flute"
      \voiceOne
      \key g \major
      \time 2/2
      r2 g-"Flute" ~
      g fis ~
      fis4 g8 fis e2 ~
      e4 d8 cis d2
    }
    \new Voice \relative c'' {
      \set midiInstrument = #"clarinet"
      \voiceTwo
      b1-"Clarinet"
      a2. b8 a
      g2. fis8 e
      fis2 r
    }
  >>
  \layout { }
  \midi {
    \context {
      \Staff
      \remove "Staff_performer"
    }
    \context {
      \Voice
      \consists "Staff_performer"
    }
  }
  \tempo 2 = 72
}
```



### 3.5.8 Proprietà di contesto per gli effetti MIDI

Le seguenti proprietà di contesto possono essere usate per applicare vari effetti MIDI alle note suonate sul canale MIDI associato col rigo, strumento MIDI o voce correnti (a seconda del valore della proprietà di contesto `Score.midiChannelMapping` e del contesto in cui si trova `Staff_performer`; vedi `<undefined>` [MIDI channel mapping], pagina `<undefined>`).

La modifica di queste proprietà di contesto avrà effetto su tutte le note suonate sul canale dopo il cambio, tuttavia alcuni effetti potrebbero essere applicati anche a note che stanno già suonando (a seconda di come è stato implementato il dispositivo MIDI).

Sono supportate le seguenti proprietà di contesto:

#### `Staff.midiPanPosition`

La posizione del panning regola il modo in cui il suono in un canale MIDI è distribuito tra gli altoparlanti stereo di sinistra e di destra. Questa proprietà di contesto accetta un numero compreso tra -1.0 (`#LEFT`) e 1.0 (`#RIGHT`); il valore -1.0 sposterà tutto il suono sull'altoparlante sinistro (rendendo muto quello destro), il valore 0.0 (`#CENTER`) distribuirà il suono in modo uniforme tra gli altoparlanti sinistro e destro, e il valore 1.0 sposterà tutto il suono sull'altoparlante destro. I valori intermedi tra -1.0 e 1.0 permettono di ottenere distribuzioni miste tra gli altoparlanti sinistro e destro.

#### `Staff.midiBalance`

Il bilanciamento stereo di un canale MIDI. In modo simile al panning, questa proprietà di contesto accetta un numero compreso tra -1.0 (`#LEFT`) e 1.0 (`#RIGHT`). Varia il volume relativo inviato ai due altoparlanti stereo senza alterare la distribuzione dei segnali stereo.

#### `Staff.midiExpression`

Livello dell'espressione (come frazione del livello massimo disponibile) da applicare a un canale MIDI. Un dispositivo MIDI combina insieme il livello di espressione del canale MIDI con l'attuale livello di dinamica di una voce (stabilito dai comandi `\p` o `\ff`) per ottenere il volume totale di ogni nota nella voce. Il controllo dell'espressione può essere usato, per esempio, per implementare effetti di crescendo o decrescendo su singole note sostenute (non supportato automaticamente da LilyPond).

La gamma del livello di espressione va da 0.0 (nessuna espressione, ovvero volume zero) a 1.0 (piena espressione).

#### `Staff.midiReverbLevel`

Livello di riverbero (come frazione del livello massimo disponibile) da applicare a un canale MIDI. Questa proprietà accetta numeri compresi tra 0.0 (nessun riverbero) e 1.0 (pieno effetto).

#### `Staff.midiChorusLevel`

Livello del chorus (come frazione del livello massimo disponibile) da applicare a un canale MIDI. Questa proprietà accetta numeri compresi tra 0.0 (nessun effetto chorus) e 1.0 (pieno effetto).

### Problemi noti e avvertimenti

Dato che i file MIDI non contengono veri dati audio, le modifiche in queste proprietà di contesto si traducono soltanto in richieste di modifica dei controlli del canale MIDI nei file MIDI generati. Se un particolare dispositivo MIDI (come un lettore MIDI) sia in grado di gestire queste richieste dipende interamente dall'implementazione del dispositivo: un dispositivo potrebbe scegliere di ignorare alcune o tutte le richieste. Dipende dall'implementazione del dispositivo MIDI anche il modo in cui un dispositivo MIDI interpreta valori diversi di questi controlli (generalmente lo

standard MIDI corregge il comportamento solo alle estremità della gamma di valori disponibile per ogni controllo) e se un cambiamento nel valore di un controllo avrà effetto anche su note che stanno già suonando su quel canale MIDI.

Quando genera file MIDI, LilyPond trasformerà semplicemente le frazioni comprese in ciascuna gamma in valori di una gamma di numeri interi corrispondenti (0-127 per i 7-bit, 0-32767 per i 14 bit per i controlli del canale MIDI che supportano una migliore risoluzione), arrotondando i valori delle frazioni sul numero intero più vicino. I valori interi convertiti sono salvati così come sono nel file MIDI generato. Consultare la documentazione del proprio dispositivo MIDI per sapere come il dispositivo interpreta questi valori.

### 3.5.9 Miglioramento dell'output MIDI

L'output MIDI predefinito fornisce le funzionalità basilari, ma può essere migliorato impostando gli strumenti MIDI, le proprietà del blocco `\midi` e/o usando lo script `articulate`.

#### Lo script `articulate`

Per usare lo script `articulate` aggiungere il relativo comando `\include` all'inizio del file di input:

```
\include "articulate.ly"
```

Lo script crea file MIDI in cui la durata delle note è correttamente “ridimensionata” per rappresentare molte articolazioni e indicazioni di tempo. Tuttavia anche l'output grafico verrà modificato per corrispondere esattamente all'output MIDI.

```
\score {
  \articulate <<
    ... musica ...
  >>
  \midi { }
}
```

Il comando `\articulate` permette di elaborare le abbreviazioni (come i trilli e i gruppetti). L'elenco completo degli elementi supportati si trova nello script stesso. Vedi `ly/articulate.ly`.

#### Vedi anche

Manuale di apprendimento: Sezione “Altre fonti di informazione” in *Manuale di Apprendimento*.

Guida alla notazione: `<undefined>` [Score layout], pagina `<undefined>`.

File installati: `ly/articulate.ly`.

**Nota:** Lo script `articulate` potrebbe accorciare la durata degli accordi, cosa che potrebbe non essere appropriata per alcuni tipi di strumento, come nel caso della musica per organo. Potrebbe essere ridotta anche la durata di note prive di articolazioni. Dunque per compensare tutto ciò, limitare l'uso del comando `\articulate` a brevi segmenti di musica o modificare i valori delle variabili definite nello script `articulate`.

## 3.6 Estrazione dell'informazione musicale

Oltre a creare l'output grafico e il MIDI, LilyPond può mostrare l'informazione musicale in forma testuale.

### 3.6.1 Mostrare la notazione LilyPond

La funzione musicale `\displayLilyMusic` permette di mostrare un'espressione musicale nella notazione di LilyPond. Per vedere l'output, di norma si esegue LilyPond da linea di comando. Per esempio,

```
{
  \displayLilyMusic \transpose c a, { c4 e g a bes }
}

mostrerà

{ a,4 cis4 e4 fis4 g4 }
```

LilyPond stampa questi messaggi nella console insieme a tutti gli altri messaggi della compilazione. Per separare questi messaggi e salvare i risultati di `\displayLilyMusic`, reindirizzare l'output su un file:

```
lilypond file.ly >display.txt
```

Nota bene che Lilypond non soltanto mostra in console l'espressione musicale, ma la interpreta anche (infatti l'espressione musicale di `\displayLilyMusic` appare nell'output oltre a essere mostrata in console). Ciò è comodo, perché si può inserire `\displayLilyMusic` in file esistenti, senza doverne modificare le note di input, per ottenere informazioni su di essi. Se non si vuole che Lilypond interpreti la musica mostrata in console, basta precedere il comando con `\void` per far sì che sia ignorata:

```
{
  \void \displayLilyMusic \transpose c a, { c4 e g a bes }
}
```

### 3.6.2 Mostrare le espressioni musicali scheme

Vedi Sezione “Displaying music expressions” in *Estendere*.

### 3.6.3 Salvare eventi musicali in un file

Gli eventi musicali possono essere salvati in file diversi, un file per ogni rigo, includendo un file nella partitura principale:

```
\include "event-listener.ly"
```

Così verranno creati uno o più file `NOMEFILE-NOMERIGO.notes` o `NOMEFILE-unnamed-staff.notes` per ogni rigo. Se ci sono vari rigi senza nome, gli eventi di tutti i rigi saranno mischiati insieme nello stesso file. I file hanno una struttura di questo tipo:

```
0.000  note      57      4  p-c 2 12
0.000  dynamic   f
0.250  note      62      4  p-c 7 12
0.500  note      66      8  p-c 9 12
0.625  note      69      8  p-c 14 12
0.750  rest      4
0.750  breathe
```

La sintassi prevede una riga delimitata da tabulatori, con due campi fissi su ogni riga seguiti da parametri opzionali.

```
tempo tipo ...parametri...
```

Questa informazione può essere letta facilmente da altri programmi, per esempio da uno script python, e può essere molto utile per ricercatori che desiderano fare delle analisi musicali o degli esperimenti di *playback* con LilyPond.

## Problemi noti e avvertimenti

Non tutti gli eventi musicali di lilypond sono supportati da `event-listener.ly`, che vuole essere una semplice “prova di concetto” ben congeniata. Se alcuni eventi che si vorrebbero vedere non sono inclusi, copiare `event-listener.ly` nella propria directory lilypond e modificare il file in modo che produca l’informazione desiderata.



## 4 Gestione dello spazio

La formattazione globale del foglio è determinata da tre fattori: la formattazione della pagina, le interruzioni di linea e la spaziatura. Ciascun fattore influenza l'altro. La scelta della spaziatura determina la densità con cui vengono disposti i sistemi musicali, che a sua volta influenza la scelta delle interruzioni di linea e quindi infine quante pagine occupa un brano.

Generalmente, questa procedura si svolge in quattro stadi. Inizialmente vengono scelte le distanze flessibili ('springs'), in base alle durate. Poi vengono tentate tutte le possibili combinazioni per le interruzioni di linea e viene calcolato un punteggio 'negativo' per ciascuna di esse. Quindi viene calcolata l'altezza di ogni possibile sistema. Infine viene scelta una combinazione di interruzioni di linea e di interruzioni di pagina che assicuri che la spaziatura verticale e quella orizzontale non siano né troppo compresse né troppo allungate.

Due tipi di blocchi possono contenere le impostazioni di formattazione: `\paper {...}` e `\layout {...}`. Il blocco `\paper` contiene le impostazioni di formattazione della pagina da applicare a tutte le partiture di un libro o di una parte, come l'altezza del foglio o se stampare o meno i numeri di pagina, etc. (vedi [\[Page layout\]](#), pagina [\(undefined\)](#)). Il blocco `\layout` contiene le impostazioni di formattazione della partitura, come il numero di sistemi da usare o lo spazio tra i gruppi di righe, etc. (vedi [\[Score layout\]](#), pagina [\(undefined\)](#)).

### 4.1 Formattazione della pagina

Questa sezione tratta le opzioni di formattazione della pagina per il blocco `\paper`.

#### 4.1.1 Il blocco `\paper`

I blocchi `\paper` possono essere posizionati in tre punti diversi, in modo da formare una gerarchia discendente di blocchi `\paper`:

- All'inizio del file di input, prima di tutti i blocchi `\book`, `\bookpart` e `\score`.
- Dentro un blocco `\book` ma fuori da tutti i blocchi `\bookpart` e `\score` in esso racchiusi.
- Dentro un blocco `\bookpart` ma fuori da tutti i blocchi `\score` in esso racchiusi.

Non è possibile inserire un blocco `\paper` in un blocco `\score`.

I valori dei campi vengono filtrati dall'alto verso il basso attraverso questa gerarchia: i valori impostati nei livelli più alti della gerarchia persistono finché non sono sovrascritti da un valore impostato in un livello più basso.

Vari blocchi `\paper` possono apparire in ognuno di questi livelli, per esempio come parti di vari file inclusi con `\include`. In questo caso, i campi di ciascun livello vengono combinati e i valori riscontrati per ultimi avranno la precedenza in caso di campi duplicati.

Le impostazioni che possono apparire in un blocco `\paper` comprendono:

- la funzione scheme `set-paper-size`,
- le variabili `\paper` per personalizzare la formattazione della pagina e
- le definizioni di markup usate per personalizzare la formattazione di intestazioni, piè di pagina e titoli.

La funzione `set-paper-size` è trattata nella prossima sezione, [\(undefined\)](#) [Paper size and automatic scaling], pagina [\(undefined\)](#). Le variabili `\paper` che si occupano della formattazione della pagina sono trattate in sezioni successive. Le definizioni di markup relative a intestazioni, piè di pagina e titoli sono trattate in [\(undefined\)](#) [Custom titles headers and footers], pagina [\(undefined\)](#).

La maggior parte delle variabili `\paper` funzionano soltanto in un blocco `\paper`. Le poche che funzionano anche in un blocco `\layout` sono elencate in [\(undefined\)](#) [Il blocco `\layout`], pagina [\(undefined\)](#).

Se non indicato diversamente, tutte le variabili `\paper` che corrispondono a distanze sulla pagina sono misurate in millimetri, a meno che un'unità di misura diversa non sia specificata dall'utente. Per esempio, la seguente dichiarazione imposta `top-margin` su dieci millimetri:

```
\paper {
  top-margin = 10
}
```

Per impostarla su 0.5 pollici, usare il suffisso di unità `\in` (inch = pollice):

```
\paper {
  top-margin = 0.5\in
}
```

I suffissi di unità disponibili sono `\mm`, `\cm`, `\in` e `\pt`. Queste unità sono semplici valori utili per convertire dai millimetri e sono definite in `ly/paper-defaults-init.ly`. Solo per chiarezza, quando si usano i millimetri, di solito si usa il suffisso `\mm`, anche se non è tecnicamente necessario.

È anche possibile definire i valori di `\paper` con Scheme. L'equivalente Scheme dell'esempio precedente è:

```
\paper {
  #(define top-margin (* 0.5 in))
}
```

## Vedi anche

Guida alla notazione: [\[Paper size and automatic scaling\]](#), pagina [\[Custom titles headers and footers\]](#), pagina [\[Il blocco \layout\]](#), pagina [\[Automatic scaling to paper size\]](#).

File installati: `ly/paper-defaults-init.ly`.

### 4.1.2 Formato carta e ridimensionamento automatico

#### Impostare il formato carta

'A4' è il valore predefinito quando non viene impostato esplicitamente alcun formato carta. Esistono due funzioni che permettono di cambiare formato:

```
set-default-paper-size
  #(set-default-paper-size "quarto")
  che deve sempre trovarsi nel livello superiore, e

set-paper-size
  \paper {
    #(set-paper-size "tabloid")
  }
  che deve sempre trovarsi in un blocco \paper.
```

La funzione `set-default-paper-size`, se usata nel livello superiore, deve precedere qualsiasi blocco `\paper`. `set-default-paper-size` imposta il formato di tutte le pagine, mentre `set-paper-size` imposta il formato soltanto di quelle pagine a cui è applicato il blocco `\paper`. Per esempio, se il blocco `\paper` si trova all'inizio del file, applicherà il formato a tutte le pagine. Se il blocco `\paper` si trova dentro un blocco `\book`, il formato verrà applicato a quel libro soltanto.

Quando si usa la funzione `set-paper-size`, questa deve essere posta *prima* di qualsiasi altra funzione usata nello stesso blocco `\paper`. Vedi [\[Automatic scaling to paper size\]](#), pagina [\[Automatic scaling to paper size\]](#).

I formati carta sono definiti in `scm/paper.scm`, e sebbene sia possibile aggiungere formati personalizzati in questo file, tali aggiunte verrebbero sovrascritte da successivi aggiornamenti del software. I formati disponibili sono elencati in `<undefined>` [Predefined paper sizes], pagina `<undefined>`.

È tuttavia possibile aggiungere un formato personalizzato nel file di input per poi utilizzarlo con `set-default-paper-size` o `set-paper-size`:

```
#(set! paper-alist (cons '("mio formato" . (cons (* 15 in) (* 3 in))) paper-alist))

\paper {
  #(set-paper-size "mio formato")
}
```

Si può usare qualsiasi unità di misura: `in` (inch, o pollici), `cm` (centimetri) e `mm` (millimetri).

Aggiungendo il simbolo `'landscape` alla funzione del formato, le pagine vengono ruotate di 90 gradi e le linee occupano il maggior spazio a disposizione.

```
#(set-default-paper-size "a6" 'landscape)
```

Appendendo `'landscape` (orizzontale) al nome del formato, è possibile scambiare le dimensioni della carta *senza* ruotare la stampa (come quando si stampa in formato cartolina o si creano dei file grafici da includere invece di un documento indipendente):

```
#(set-default-paper-size "a6landscape")
```

Quando il formato termina con un esplicito `'landscape` (orizzontale) o `'portrait` (verticale), la presenza di un simbolo `'landscape` influisce *solo* sull'orientamento della stampa, non sul formato usato per la formattazione.

## Vedi anche

Guida alla notazione: `<undefined>` [Automatic scaling to paper size], pagina `<undefined>`, `<undefined>` [Predefined paper sizes], pagina `<undefined>`.

File installati: `scm/paper.scm`.

## Ridimensionamento automatico al formato carta

Se il formato viene cambiato con una delle funzioni scheme (`set-default-paper-size` o `set-paper-size`), i valori di diverse variabili `\paper` sono automaticamente ricalcolati in base alla nuova dimensione. Per aggirare il ridimensionamento automatico di una certa variabile, impostare la variabile dopo aver impostato il formato. Fare attenzione al fatto che il ridimensionamento automatico non viene attivato se si impostano le variabili `paper-height` o `paper-width`, anche se `paper-width` può influenzare altri valori (ma questo è un argomento diverso dal ridimensionamento ed è trattato in seguito). Le funzioni `set-default-paper-size` e `set-paper-size` sono descritte in `<undefined>` [Setting the paper size], pagina `<undefined>`.

Le dimensioni verticali interessate dal ridimensionamento automatico sono `top-margin` e `bottom-margin` (vedi `<undefined>` [Variabili fisse della spaziatura verticale di `\paper`], pagina `<undefined>`). Le dimensioni orizzontali interessate dal ridimensionamento automatico sono `left-margin`, `right-margin`, `inner-margin`, `outer-margin`, `binding-offset`, `indent` e `short-indent` (vedi `<undefined>` [Variabili della spaziatura orizzontale di `\paper`], pagina `<undefined>`).

I valori predefiniti di queste dimensioni sono impostati in `ly/paper-defaults-init.ly` e salvati in variabili interne chiamate `top-margin-default`, `bottom-margin-default`, etc. Questi valori si riferiscono al formato predefinito `a4`. Per riferimento, nel formato `a4` il valore di `paper-height` è 297\mm e quello di `paper-width` è 210\mm.

## Vedi anche

Guida alla notazione: `<undefined>` [Variabili fisse della spaziatura verticale di `\paper`], pagina `<undefined>`, `<undefined>` [Variabili della spaziatura orizzontale di `\paper`], pagina `<undefined>`.

File installati: `ly/paper-defaults-init.ly`, `scm/paper.scm`.

### 4.1.3 Variabili `\paper` della spaziatura verticale fissa

**Nota:** Alcune dimensioni definite nel blocco `\paper` sono ridimensionate automaticamente in base al formato carta, portando a possibili risultati imprevisti. Vedi `<undefined>` [Automatic scaling to paper size], pagina `<undefined>`.

I valori predefiniti (prima del ridimensionamento) sono definiti in `ly/paper-defaults-init.ly`.

#### `paper-height`

L'altezza della pagina (valore predefinito: non impostata). Il ridimensionamento automatico di alcune dimensioni verticali non è influenzato da questa.

#### `top-margin`

Il margine tra il punto più alto della pagina e il punto più alto dell'area di stampa. Se il formato viene modificato, il valore predefinito di questa dimensione viene ridimensionato di conseguenza.

#### `bottom-margin`

Il margine tra il punto più basso dell'area di stampa e il punto più basso della pagina. Se il formato viene modificato, il valore predefinito di questa dimensione viene ridimensionato di conseguenza.

#### `ragged-bottom`

Se impostato su vero, i sistemi avranno la loro naturale spaziatura. Non saranno giustificati, ovvero non saranno né compressi né allungati verticalmente per stare nella pagina.

#### `ragged-last-bottom`

Se impostato su falso, l'ultima pagina, e l'ultima pagina di ogni sezione creata con un blocco `\bookpart`, saranno giustificate verticalmente come nelle pagine precedenti.

## Vedi anche

Guida alla notazione: `<undefined>` [Automatic scaling to paper size], pagina `<undefined>`.

File installati: `ly/paper-defaults-init.ly`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

## Problemi noti e avvertimenti

I titoli (definiti nel blocco `\header`) sono considerati un sistema, dunque `ragged-bottom` e `ragged-last-bottom` aggiungeranno spazio tra i titoli e il primo sistema della partitura.

Formati carta definiti esplicitamente sovrascriveranno qualsiasi impostazione di margine superiore o inferiore definito dall'utente.

### 4.1.4 Variabili `\paper` della spaziatura verticale flessibile

Nella maggior parte dei casi è preferibile che le distanze verticali tra certi elementi (come margini, titoli, sistemi e partiture separate) siano flessibili, così che siano ben compresse o allungate a seconda della situazione. Sono disponibili diverse variabili `\paper` (elencate sotto) per regolare il comportamento di allungamento di queste dimensioni.

Nota bene che le variabili `\paper` trattate in questa sezione non regolano la spaziatura dei rigli dei singoli sistemi. La spaziatura interna ai sistemi è controllata dalle proprietà del grob, solitamente inserite in un blocco `\score` o `\layout` e non in un blocco `\paper`. Vedi `<undefined>` [Flexible vertical spacing within systems], pagina `<undefined>`.

## Struttura delle liste associative flessibili della spaziatura verticale

Ognuna delle variabili `\paper` flessibili della spaziatura verticale è una lista associativa (alist) composta da quattro *elementi*:

- **basic-distance** – la distanza verticale, misurata in spazi rigo, tra i *punti di riferimento* dei due elementi, quando non si verificano collisioni e non sono attivi allungamenti o compressioni. Il punto di riferimento di un markup (un titolo o un markup di un livello superiore) è il suo punto più alto, mentre quello di un sistema è il centro verticale del più vicino `StaffSymbol`, anche se una linea diversa da un rigo (come un contesto `Lyrics`) si trova nel mezzo. Valori di **basic-distance** inferiori a quelli di **padding** o **minimum-distance** non hanno senso, dato che la distanza risultante non sarà mai inferiore a quella definita per **padding** o **minimum-distance**.
- **minimum-distance** – la minima distanza verticale consentita, misurata in spazi rigo, tra i punti di riferimento dei due elementi, quando la compressione è attiva. Valori di **minimum-distance** inferiori a quelli di **padding** non hanno senso, perché la distanza risultante non sarà mai inferiore a quella definita per **padding**.
- **padding** – la quantità minima di spazio verticale libero tra i profili (*skyline*) dei due elementi, misurata in spazi rigo.
- **stretchability** – una misura, priva di unità di misura, della relativa propensione della dimensione a allungarsi. Se pari a zero, la distanza non si allungherà (a meno che non ci siano delle collisioni). Se positiva, il significato del valore di **stretchability** di una certa dimensione sta nella sua relazione con i valori di **stretchability** delle altre dimensioni. Per esempio, se una dimensione ha un valore di **stretchability** doppio rispetto a un'altra, si allungherà il doppio. I valori devono essere non negativi e finiti. Il valore `+inf.0` causa un errore di programmazione e viene ingorato, ma si può usare `1.0e7` per ottenere un'elasticità allungabile quasi all'infinito. Se non impostata, il valore predefinito equivale a quello di **basic-distance**. Nota che la propensione della dimensione a *comprimersi* non può essere impostata direttamente dall'utente ed è uguale a  $(\text{basic-distance} - \text{minimum-distance})$ .

Se in una pagina lo spazio verticale non è giustificato (ovvero con **ragged-bottom** impostato su vero) la distanza risultante è data dal valore più grande assegnato a una di queste tre proprietà:

- **basic-distance**,
- **minimum-distance** e
- **padding** con l'aggiunta della più piccola distanza necessaria a eliminare collisioni.

In caso di partiture con molte pagine in cui l'ultima pagina non è giustificata (ovvero con **ragged-last-bottom** impostato su vero), l'ultima pagina usa la stessa spaziatura della pagina precedente, purché ci sia abbastanza spazio.

I metodi specifici per modificare le liste associative (alist) sono trattati in Sezione 5.3.6 [Modifying alists], pagina 608. L'esempio seguente illustra i due modi in cui queste possono essere modificate. La prima dichiarazione aggiorna un elemento-valore individualmente, mentre la seconda ridefinisce completamente la variabile:

```
\paper {
  system-system-spacing.basic-distance = #8
  score-system-spacing =
    #'(basic-distance . 12)
      (minimum-distance . 6)
```

```
(padding . 1)
(stretchability . 12))
}
```

## Elenco delle variabili `\paper` flessibili della spaziatura verticale

I nomi di queste variabili hanno il formato *superiore-inferiore-spacing*, dove *superiore* e *inferiore* indicano gli elementi di cui stabilire la distanza. Ogni distanza viene misurata tra i punti di riferimento dei due elementi (vedi la descrizione precedente della struttura della lista associativa). Nei nomi di queste variabili il termine ‘markup’ si riferisce sia ai *titoli* (`bookTitleMarkup` o `scoreTitleMarkup`) sia ai *markup di livello superiore* (vedi `\undefined` [File structure], pagina `\undefined`). Tutte le distanze sono misurate in spazi rigo.

Le impostazioni predefinite sono definite in `ly/paper-defaults-init.ly`.

### `markup-system-spacing`

la distanza tra il testo (titolo o markup di livello superiore) e il sistema che lo segue.

### `score-markup-spacing`

la distanza tra l’ultimo sistema di un brano e il testo (titolo o markup di livello superiore) che lo segue.

### `score-system-spacing`

la distanza tra l’ultimo sistema di un brano e il primo sistema del brano successivo, quando non c’è alcun testo (titolo o markup di livello superiore) tra di loro.

### `system-system-spacing`

la distanza tra due sistemi di uno stesso brano.

### `markup-markup-spacing`

la distanza tra due testi (titoli o markup di livello superiore).

### `last-bottom-spacing`

la distanza tra l’ultimo sistema o l’ultimo markup di livello superiore di una pagina e la fine dello spazio stampabile (ovvero la parte superiore del margine inferiore).

### `top-system-spacing`

la distanza tra l’inizio dello spazio stampabile (ovvero la parte inferiore del margine superiore) e il primo sistema di una pagina, quando non c’è alcun testo (titolo o markup di livello superiore) tra di loro.

### `top-markup-spacing`

la distanza tra l’inizio dello spazio stampabile (ovvero la parte inferiore del margine superiore) e il primo testo (titolo o markup di livello superiore) di una pagina, quando non c’è alcun sistema tra i due.

## Vedi anche

Guida alla notazione: `\undefined` [Flexible vertical spacing within systems], pagina `\undefined`.

File installati: `ly/paper-defaults-init.ly`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

## 4.1.5 Variabili `\paper` della spaziatura orizzontale

**Nota:** Alcune dimensioni in `\paper` sono ridimensionate automaticamente in base al formato carta, producendo talvolta un risultato inatteso. Vedi `\undefined` [Automatic scaling to paper size], pagina `\undefined`.

## Variabili `\paper` per larghezze e margini

I valori predefiniti (prima del ridimensionamento) non elencati qui si trovano in `ly/paper-defaults-init.ly`.

### `paper-width`

Larghezza della pagina (non impostata inizialmente). `paper-width`, pur non avendo effetto sul ridimensionamento automatico di alcune dimensioni orizzontali, influenza la variabile `line-width`. Se sono impostate sia `paper-width` che `line-width`, saranno modificate anche `left-margin` e `right-margin`. Vedi anche `check-consistency`.

### `line-width`

Se specificata in un blocco `\paper` definisce la lunghezza orizzontale disponibile per i righi musicali nei sistemi non indentati. Se non specificata, la larghezza della linea (`line-width`) del foglio è determinata dalla sottrazione (`paper-width` - `left-margin` - `right-margin`). Se `line-width` è specificata nel blocco `\paper` e entrambi i margini, `left-margin` e `right-margin`, non lo sono, questi saranno aggiornati in modo da centrare i sistemi sulla pagina automaticamente. Vedi anche `check-consistency`.

Le variabili `line-width` per i singoli brani possono essere specificate nel blocco `\layout` di ogni brano, ovvero dentro un blocco `\score`. Questi valori regolano la larghezza delle linee prodotte brano per brano. Se `line-width` non viene specificata per un brano, il suo valore predefinito coincide con la larghezza della linea del foglio, ovvero col valore della variabile `line-width` del blocco `\paper`. Impostando `line-width` per un brano in particolare, ovvero all'interno di un blocco `\score`, i margini del foglio non vengono modificati. Le linee del rigo, la cui lunghezza è determinata dalla variabile `line-width` del brano, sono allineate a sinistra all'interno dello spazio definito dalla variabile `line-width` del foglio. Se le variabili `line-width` del brano e del foglio sono uguali, le linee del rigo si estenderanno esattamente dal margine sinistro al margine destro, ma se quella del brano è maggiore di quella del foglio le linee del rigo andranno oltre il margine destro.

### `left-margin`

Margine tra il bordo sinistro della pagina e l'inizio delle linee del rigo nei sistemi non indentati. Se il formato è modificato, il valore predefinito di questa dimensione viene ridimensionato proporzionalmente. Se `left-margin` non è impostato, e sono impostati sia `line-width` che `right-margin`, allora `left-margin` equivale alla differenza (`paper-width` - `line-width` - `right-margin`). Se è impostata soltanto `line-width`, entrambi i margini sono impostati su  $((\text{paper-width} - \text{line-width}) / 2)$ , e di conseguenza i sistemi sono centrati sulla pagina. Vedi anche `check-consistency`.

### `right-margin`

Margine tra il bordo destro della pagina e la fine delle linee del rigo in sistemi giustificati. Se il formato è modificato, il valore predefinito di questa dimensione viene ridimensionato proporzionalmente. Se `right-margin` non è impostato, e sono impostati sia `line-width` che `left-margin`, allora `right-margin` equivale a (`paper-width` - `line-width` - `left-margin`). Se è impostata soltanto `line-width`, entrambi i margini sono impostati su  $((\text{paper-width} - \text{line-width}) / 2)$ , e di conseguenza i sistemi sono centrati sulla pagina. Vedi anche `check-consistency`.

### `check-consistency`

Se impostata su vero (valore predefinito), appare un avvertimento nella console se la somma di `left-margin`, `line-width` e `right-margin` non corrisponde esatta-

mente a `paper-width`, e a ciascuna di queste variabili (eccetto `paper-width`) sarà assegnato il suo valore predefinito (ridimensionato in base al formato, se richiesto). Se impostato su falso, qualsiasi incongruenza viene ignorata e i sistemi potrebbero andare oltre il bordo della pagina.

#### `ragged-right`

Se impostato su vero, i sistemi non occuperanno interamente la larghezza della linea, ovvero non saranno giustificati. Termineranno invece alla loro lunghezza orizzontale naturale. Valori predefiniti: `#t` (true = vero) per i brani con un unico sistema, e `#f` (false = falso) per i brani con due o più sistemi. Questa variabile può essere impostata anche in un blocco `\layout`.

#### `ragged-last`

Se impostata su vero, l'ultimo sistema del brano non occuperà interamente la larghezza della linea, ovvero non sarà giustificato. Terminerà invece alla sua lunghezza orizzontale naturale. Valore predefinito: `#f` (false = falso). Questa variabile può essere impostata anche in un blocco `\layout`.

### Vedi anche

Guida alla notazione: `<undefined>` [Automatic scaling to paper size], pagina `<undefined>`.

File installati: `ly/paper-defaults-init.ly`.

### Problemi noti e avvertimenti

Formati carta definiti esplicitamente sovrascriveranno qualsiasi margine destro o sinistro definito dall'utente.

### Variabili `\paper` per la modalità due pagine per foglio

I valori predefiniti (prima del ridimensionamento) sono definiti in `ly/paper-defaults-init.ly`.

#### `two-sided`

Se impostata su vero, vengono usati `inner-margin`, `outer-margin` e `binding-offset` per determinare i margini a seconda che il numero di pagina sia dispari o pari. Questa variabile sovrascrive `left-margin` e `right-margin`.

#### `inner-margin`

Margine del lato interno di tutte le pagine se queste fanno parte di un libro. Se il formato è modificato, il valore predefinito di questa dimensione viene ridimensionato proporzionalmente. Funziona soltanto con `two-sided` impostato su vero.

#### `outer-margin`

È il margine del lato esterno di tutte le pagine se queste fanno parte di un libro. Se il formato è modificato, il valore predefinito di questa dimensione viene ridimensionato proporzionalmente. Funziona soltanto con `two-sided` impostato su vero.

#### `binding-offset`

Quantità di spazio da aggiungere a `inner-margin` per assicurarsi che niente sia nascosto dalla rilegatura. Se il formato è modificato, il valore predefinito di questa dimensione viene ridimensionato proporzionalmente. Funziona soltanto con `two-sided` impostato su vero.

### Vedi anche

Guida alla notazione: `<undefined>` [Automatic scaling to paper size], pagina `<undefined>`.

File installati: `ly/paper-defaults-init.ly`.



## Variabili `\paper` per spostamenti e indentazioni

I valori predefiniti (prima del ridimensionamento) non elencati qui sono visibili in `ly/paper-defaults-init.ly`.

### `horizontal-shift`

È la quantità di spazio di cui spostare a destra tutti i sistemi (inclusi i titoli e i separatori dei sistemi). Valore predefinito: `0.0\mm`.

### `indent`

Livello di indentazione del primo sistema di un brano. Se il formato è modificato, il valore predefinito di questa dimensione viene ridimensionato proporzionalmente. Lo spazio compreso in `line-width` disponibile per il primo sistema è ridotto di questa quantità. `indent` può essere specificato anche in blocchi `\layout` per impostare l'indentazione brano per brano.

### `short-indent`

Livello di indentazione di tutti i sistemi di un brano eccetto il primo sistema. Se il formato è modificato, il valore predefinito di questa dimensione viene ridimensionato proporzionalmente. Lo spazio compreso in `line-width` disponibile per tutti i sistemi tranne il primo è ridotto di questa quantità. `short-indent` può essere specificato anche in blocchi `\layout` per impostare le brevi indentazioni brano per brano.

## Vedi anche

Guida alla notazione: `<undefined>` [Automatic scaling to paper size], pagina `<undefined>`.

File installati: `ly/paper-defaults-init.ly`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

## 4.1.6 Altre variabili di `\paper`

### Variabili di `\paper` per l'interruzione di linea

#### `max-systems-per-page`

Il numero massimo di sistemi che saranno disposti in una pagina. Attualmente ciò è supportato soltanto dall'algoritmo `ly:optimal-breaking`. Valore predefinito: non impostato.

#### `min-systems-per-page`

Il numero minimo di sistemi che saranno disposti in una pagina. Ciò potrebbe riempire troppo le pagine, se il numero è troppo grande. Attualmente ciò è supportato soltanto dall'algoritmo `ly:optimal-breaking`. Valore predefinito: non impostato.

#### `systems-per-page`

Il numero di sistemi da disporre in una pagina. Attualmente ciò è supportato soltanto dall'algoritmo `ly:optimal-breaking`. Valore predefinito: non impostato.

#### `system-count`

Il numero di sistemi da usare per un brano. Valore predefinito: non impostato. Questa variabile può essere impostata anche in un blocco `\layout`.

## Vedi anche

Guida alla notazione: `<undefined>` [Line breaking], pagina `<undefined>`.

## Variabili di `\paper` per l'interruzione di pagina

I valori predefiniti non elencati qui sono visibili in `ly/paper-defaults-init.ly`

### `page-breaking`

L'algoritmo di interruzione di pagina da usare. Le opzioni sono `ly:minimal-breaking`, `ly:page-turn-breaking`, `ly:one-line-breaking` e `ly:optimal-breaking` (predefinito).

### `page-breaking-system-system-spacing`

Induce il sistema di interruzione di pagina a credere che `system-system-spacing` sia impostato in modo diverso dal modo in cui è impostato realmente. Per esempio, se `page-breaking-system-system-spacing #'padding` è impostato su un valore molto più grande di `system-system-spacing #'padding`, il sistema di interruzione di pagina disporrà meno sistemi su ciascuna pagina. Valore predefinito: non impostato.

### `page-count`

Il numero di pagine da usare per un brano, non impostato.

Le variabili seguenti sono attive soltanto quando `page-breaking` è impostato su `ly:page-turn-breaking`. Le interruzioni di pagina vengono quindi scelte per minimizzare il numero di voltate di pagina. Dato che le voltate di pagina si rendono necessarie quando si passa da una pagina con numero dispari a una con numero pari, verrà solitamente favorita una formattazione in cui l'ultima pagina abbia un numero dispari. I punti preferiti per le voltate di pagina possono essere indicati manualmente col comando `\allowPageTurn` o automaticamente includendo l'incisore `Page_turn_engraver` (vedi `<undefined>` [Optimal page turning], pagina `<undefined>`).

Se le scelte disponibili per avere voltate di pagina adeguate sono insufficienti, LilyPond potrebbe inserire una pagina vuota all'interno di un brano o tra i brani (se ce ne sono almeno due), oppure potrebbe terminare un brano su una pagina pari. Aumentando i valori delle seguenti tre variabili diminuisce la probabilità che queste entrino in azione.

I valori sono penalità: maggiore il valore, minore la probabilità che si verifichi l'azione associata rispetto alle altre scelte.

### `blank-page-penalty`

La penalità per avere una pagina vuota nel mezzo di un brano. Se il valore di `blank-page-penalty` è grande ed è selezionato `ly:page-turn-breaking`, sarà meno probabile che LilyPond inserisca una pagina nel mezzo del brano. Aggiungerà invece maggior spazio tra i sistemi per occupare la pagina vuota e quella successiva. Valore predefinito: 5.

### `blank-last-page-penalty`

La penalità per terminare il brano su una pagina pari. Se il valore di `blank-last-page-penalty` è grande ed è selezionato `ly:page-turn-breaking`, sarà meno probabile che LilyPond produca una partitura in cui l'ultima pagina sia pari. Regolerà invece la spaziatura in modo da usare una pagina in più o in meno. Valore predefinito: 0.

### `blank-after-score-page-penalty`

La penalità per avere una pagina vuota dopo la fine di un brano e prima di quello successivo. Il valore predefinito è inferiore a `blank-page-penalty`, in modo che siano inserite preferibilmente pagine vuote al termine di un brano piuttosto che nel mezzo di un brano. Valore predefinito: 2.

## Vedi anche

Guida alla notazione: `<undefined>` [Page breaking], pagina `<undefined>`, `<undefined>` [Optimal page breaking], pagina `<undefined>`, `<undefined>` [Optimal page turning], pagina `<undefined>`, `<undefined>` [Minimal page breaking], pagina `<undefined>`, `<undefined>` [One-line page breaking], pagina `<undefined>`.

File installati: `ly/paper-defaults-init.ly`.

## Variabili di `\paper` per la numerazione delle pagine

I valori predefiniti non elencati qui sono visibili in `ly/paper-defaults-init.ly`

### `auto-first-page-number`

L'algoritmo di interruzione di pagina si comporta diversamente a seconda che il numero della prima pagina sia dispari o pari. Se questa variabile è impostata su vero, l'algoritmo di interruzione di pagina deciderà se iniziare con un numero dispari o un numero pari. Quindi il numero della prima pagina resterà lo stesso o aumenterà di uno. Valore predefinito: `#f` (falso).

### `first-page-number`

Il valore del numero di pagina della prima pagina.

### `print-first-page-number`

Se impostato su vero, appare il numero di pagina sulla prima pagina.

### `print-page-number`

Se impostato su falso, non appariranno i numeri di pagina.

### `page-number-type`

Il tipo di numero usato per numerare le pagine. Le opzioni possibili sono `roman-lower`, `roman-upper` e `arabic`. Valore predefinito: `'arabic`.

## Vedi anche

File installati: `ly/paper-defaults-init.ly`.

## Problemi noti e avvertimenti

I numeri di pagina dispari sono sempre sulla destra. Se la musica deve iniziare a pagina 1, ci deve essere una pagina vuota sulla seconda di copertina in modo che pagina 1 sia sul lato destro.

## Svariate variabili di `\paper`

### `page-spacing-weight`

L'importanza della spaziatura (verticale) e della linea (orizzontale) della pagina. Valori più grandi renderanno la spaziatura della pagina più importante. Valore predefinito: 10.

### `print-all-headers`

Se impostato su vero, appariranno nell'output tutte le intestazioni di ciascun blocco `\score`. Di norma appaiono soltanto le variabili `piece` e `opus`. Valore predefinito: `#f`.

### `system-separator-markup`

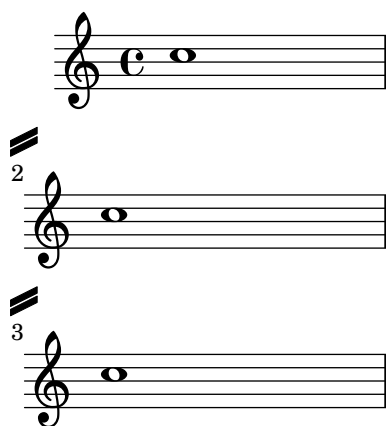
Un oggetto markup inserito tra i sistemi, spesso usato per le partiture orchestrali. Valore predefinito: non impostato. È disponibile il comando markup `\slashSeparator`, definito in `ly/titling-init.ly`, che fornisce un ragionevole valore predefinito, per esempio:

```
 #(set-default-paper-size "a8")
```

```

\book {
  \paper {
    system-separator-markup = \slashSeparator
  }
  \header {
    tagline = ##f
  }
  \score {
    \relative { c''1 \break c1 \break c1 }
  }
}

```



## Vedi anche

File installati: `ly/titling-init.ly`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

## Problemi noti e avvertimenti

L'intestazione di pagina predefinita pone sulla stessa linea il numero di pagina e il campo `instrument` del blocco `\header`.

## 4.2 Formattazione della partitura

Questa sezione tratta le opzioni di formattazione della partitura disponibili nel blocco `\layout`.

### 4.2.1 Il blocco `\layout`

Mentre il blocco `\paper` contiene le impostazioni relative alla formattazione della pagina per l'intero documento, il blocco `\layout` contiene impostazioni di formattazione specifica di un brano. Per impostare globalmente le opzioni di formattazione del brano, inserirle in un blocco `\layout` del livello superiore. Per impostare le impostazioni di formattazione per un singolo brano, inserirle in un blocco `\layout` racchiuso in un blocco `\score`, dopo la musica. Le impostazioni che possono apparire in un blocco `\layout` includono:

- la funzione `scheme layout-set-staff-size`,
- le modifiche di contesto nei blocchi `\context` e
- le variabili `\paper` che agiscono sulla formattazione del brano.

La funzione `layout-set-staff-size` è spiegata nella prossima sezione, `<undefined>` [Setting the staff size], pagina `<undefined>`. Le modifiche di contesto sono trattate in un capitolo separato;

vedi Sezione 5.1.4 [Modifying context plug-ins], pagina 587, e Sezione 5.1.5 [Changing context default settings], pagina 589.

Le variabili `\paper` che possono apparire in un blocco `\layout`, con valori predefiniti presi dal blocco `\paper`, sono:

- `line-width`, `ragged-right` e `ragged-last` (vedi `\paper` [variabili `\paper` per larghezze e margini], pagina `\paper`)
- `indent` e `short-indent` (vedi `\paper` [variabili `\paper` per spostamenti e indentazioni], pagina `\paper`)
- `system-count` (vedi `\paper` [variabili `\paper` per l'interruzione di linea], pagina `\paper`)

Ecco un esempio di blocco `\layout`:

```
\layout {
  indent = 2\cm
  \context {
    \StaffGroup
    \override StaffGroup.staff-staff-spacing.basic-distance = #8
  }
  \context {
    \Voice
    \override TextScript.padding = #1
    \override Glissando.thickness = #3
  }
}
```

Si possono inserire molteplici blocchi `\layout` come espressioni di livello superiore. Ciò può essere utile, per esempio, se impostazioni diverse sono salvate in file separati e incluse facoltativamente. Internamente, quando un blocco `\layout` viene elaborato, viene fatta una copia della configurazione del blocco `\layout` corrente, poi qualsiasi modifica apportata nel blocco viene applicata e il risultato viene salvato come la nuova configurazione corrente. Dalla prospettiva dell'utente, i blocchi `\layout` sono combinati, ma in situazioni di conflitto (ovvero quando viene modificata la stessa proprietà in blocchi diversi) hanno precedenza le definizioni più recenti.

Per esempio, se questo blocco

```
\layout {
  \context {
    \Voice
    \override TextScript.color = #magenta
    \override Glissando.thickness = #1.5
  }
}
```

viene posto dopo quello dell'esempio precedente, le sovrascritture di `'padding` e `'color` per `TextScript` sono combinate, mentre quella più recente di `'thickness` per `Glissando` sostituisce (o nasconde) quella precedente.

I blocchi `\layout` possono essere assegnati a delle variabili per poterli riusare successivamente, ma il modo in cui ciò funziona è leggermente ma significativamente diverso rispetto a scriverle davvero.

Se una variabile è definita in questo modo,

```
layoutVariable = \layout {
  \context {
    \Voice
    \override NoteHead.font-size = #4
```

```
}
}
```

manterrà la configurazione attuale di `\layout` con l'aggiunta della sovrascrittura `NoteHead.font-size`, ma questa combinazione *non* è salvata come la nuova configurazione corrente. Attenzione al fatto che la “configurazione corrente” viene letta quando la variabile è definita e non quando viene usata, dunque il contenuto della variabile dipende dalla sua posizione nel file di input.

La variabile può quindi essere usata all'interno di un altro blocco `\layout`, per esempio:

```
\layout {
  \layoutVariable
  \context {
    \Voice
    \override NoteHead.color = #red
  }
}
```

Un blocco `\layout` che contiene una variabile, come nell'esempio precedente, *non* copia la configurazione corrente bensì usa il contenuto di `\layoutVariable` come configurazione di base per altre aggiunte. Ciò significa che qualsiasi modifica definita tra la definizione della variabile e il momento in cui essa viene usata è perduta.

Se `layoutVariable` è definita (o inclusa con `\include`) subito prima di essere usata, il suo contenuto comprende soltanto la configurazione corrente più le sovrascritture definite al suo interno. Quindi nell'esempio precedente che illustra l'uso di `\layoutVariable` il blocco `\layout` finale conterrebbe:

```
TextScript.padding = #1
TextScript.color = #magenta
Glissando.thickness = #1.5
NoteHead.font-size = #4
NoteHead.color = #red

più le sovrascritture indent e StaffGrouper.
```

Ma se la variabile fosse già stata definita prima del primo blocco `\layout`, la configurazione corrente conterrebbe soltanto

```
NoteHead.font-size = #4 % (scritta nella definizione della variabile)
NoteHead.color = #red % (aggiunta dopo l'uso della variabile)
```

Se ben organizzate, le variabili `\layout` possono essere un valido strumento per strutturare le formattazioni dei sorgenti, e anche per ripristinare la configurazione di `\layout` a uno stato conosciuto.

## Vedi anche

Guida alla notazione: Sezione 5.1.5 [Changing context default settings], pagina 589.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

### 4.2.2 Impostare la dimensione del rigo

La dimensione del rigo (in inglese **staff size**) è 20 punti, che corrispondono a un'altezza del rigo di 7.03mm (un punto è uguale a 100/7227 di un pollice o a 2540/7227 mm). La dimensione del rigo può essere modificata in tre modi:

1. Per impostare la dimensione del rigo globalmente per tutti i brani di un file (o di un blocco `\book`, per essere precisi), usare `set-global-staff-size`:  

```
 #(set-global-staff-size 14)
```

L'esempio precedente imposta la dimensione del rigo globale a 14pt (4.92mm) e ridimensiona proporzionalmente tutti i tipi di carattere.

2. Per impostare la dimensione del rigo di una singola partitura in un libro, usare `layout-set-staff-size` all'interno del blocco `\layout` di quel brano:

```
\score {
  ...
  \layout {
    #(layout-set-staff-size 14)
  }
}
```

3. Per impostare la dimensione del rigo di un singolo rigo di un sistema, usare il comando `\magnifyStaff`. Per esempio, le partiture di musica da camera incise in modo tradizionale spesso usavano righe per pianoforte di 7mm mentre gli altri righe erano solitamente tra 3/5 e 5/7 più grandi (tra 60% e 71%). Per ottenere la proporzione 5/7, usare:

```
\score {
  <<
    \new Staff \with {
      \magnifyStaff #5/7
    } { ... }
    \new PianoStaff { ... }
  >>
}
```

Se si desidera una dimensione del tipo di carattere (`fontSize`) ben precisa, si può usare la seguente forma:

```
\score {
  <<
    \new Staff \with {
      \magnifyStaff #(magstep -3)
    } { ... }
    \new PianoStaff { ... }
  >>
}
```

Per emulare l'aspetto delle partiture incise coi metodi tradizionali, è meglio evitare di ridurre lo spessore delle linee del rigo.

## Corpo automatico dei tipi di carattere a dimensioni diverse

Il tipo di carattere Feta fornisce simboli musicali in otto dimensioni diverse. Ogni tipo di carattere è calibrato per una specifica dimensione del rigo: via via che la dimensione del rigo diventa più piccola, il corpo del carattere diventa più grosso, per abbinarsi alle linee del rigo più spesse. Le dimensioni del carattere consigliate sono elencate nella seguente tabella:

nome del tipo di carattere	altezza del rigo (pt)	altezza del rigo (mm)	uso
feta11	11.22	3.9	partiture tascabili
feta13	12.60	4.4	
feta14	14.14	5.0	
feta16	15.87	5.6	
feta18	17.82	6.3	canzonieri
feta20	20	7.0	parti standard
feta23	22.45	7.9	

feta26

25.2

8.9

## Vedi anche

Guida alla notazione: [\[Selecting notation font size\]](#), pagina [\[undefined\]](#).

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

## Problemi noti e avvertimenti

`layout-set-staff-size` non modifica la distanza tra le linee del rigo.

## 4.3 Interruzioni

### 4.3.1 Interruzioni di linea

Le interruzioni di linea di solito sono determinate automaticamente. Sono decise in modo che le linee non sembrino né fitte né troppo spaziate, e che le linee consecutive abbiano una densità simile.

Per forzare manualmente un'interruzione di linea dopo una stanghetta, usare il comando `\break`:

```
\relative c' {
  c4 c c c | \break
  c4 c c c |
}
```



Per impostazione predefinita, un comando `\break` nel mezzo di una misura viene ignorato e apparirà un avvertimento nella console. Per forzare un'interruzione di linea nel mezzo di una misura, aggiungere una stanghetta invisibile con `\bar ""`:

```
\relative c' {
  c4 c c
  \bar "" \break
  c |
  c4 c c c |
}
```



Un `\break` successivo a una stanghetta viene ignorato se la misura precedente termina nel mezzo di una nota, come quando un gruppo irregolare inizia e termina in misure



diverse. Per far sì che i comandi `\break` funzionino in queste situazioni, togliere l'incisore `Forbid_line_break_engraver` dal contesto `Voice`. Nota che le interruzioni di linea forzate manualmente devono essere aggiunte in parallelo con la musica:

```
\new Voice \with {
  \remove "Forbid_line_break_engraver"
} \relative {
  <<
    { c''2. \tuplet 3/2 { c4 c c } c2. | }
    { s1 | \break s1 | }
  >>
}
```



Analogamente, le interruzioni di linea sono normalmente proibite quando le travature oltrepassano una stanghetta. Tale comportamento può essere modificato impostando `\override Beam.breakable = ##t`:

```
\relative c'' {
  \override Beam.breakable = ##t
  c2. c8[ c | \break
  c8 c] c2. |
}
```



Il comando `\noBreak` vieta un'interruzione di linea sulla stanghetta in cui è inserito.

In una partitura, l'interruzione di linea automatica è vietata per la musica compresa tra i comandi `\autoLineBreaksOff` e `\autoLineBreaksOn`. Per impedire anche le interruzioni di pagina, usare i comandi `\autoBreaksOff` e `\autoBreaksOn`. Le interruzioni manuali non sono interessate da questi comandi. Nota che bloccare le interruzioni di linea automatiche potrebbe far andare la musica oltre il margine destro se questa non può essere contenuta in una linea.

Le interruzioni di linea automatiche (ma non le interruzioni di pagina) possono essere abilitate per singole stanghetta usando `\once \autoLineBreaksOn` all'inizio di una stanghetta. Ciò identifica un'interruzione di linea permessa, invece che forzata.

Le impostazioni fondamentali che influenzano la spaziatura della linea sono `indent` e `line-width`, impostate nel blocco `\layout`: regolano l'indentazione della prima linea e la lunghezza delle linee.

Se `ragged-right` è impostato su vero nel blocco `\layout`, allora i sistemi terminano alla loro naturale lunghezza orizzontale, invece di essere allungati orizzontalmente per riempire l'intera linea. Ciò è utile per brevi frammenti e per verificare quanto è stretta la spaziatura naturale.

L'opzione `ragged-last` è simile a `ragged-right`, ma agisce soltanto sull'ultima linea del brano.

```
\layout {
  indent = 0\mm
  line-width = 150\mm
  ragged-last = ##t
}
```

Per inserire interruzioni di linea a intervalli regolari usare `\break` separato da pause spaziatrici e ripetuto con `\repeat`. Per esempio, per interrompere le seguenti 28 misure (considerando un tempo di 4/4) esattamente ogni 4 misure, usare:

```
<<
\repeat unfold 7 {
  s1 \noBreak s1 \noBreak
  s1 \noBreak s1 \break
}
{ la vera musica... }
>>
```

## Comandi predefiniti

`\break`, `\noBreak`, `\autoBreaksOff`, `\autoBreaksOn`, `\autoLineBreaksOff`, `\autoLineBreaksOn`.

## Frammenti di codice selezionati

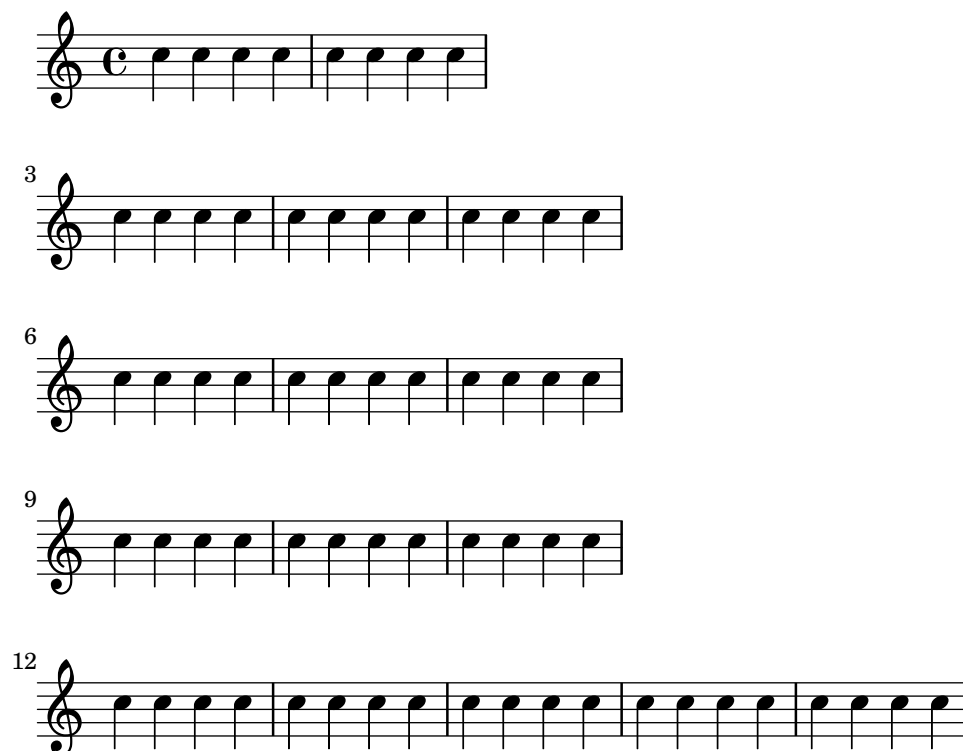
*Usare una voce apposita per le interruzioni*

Spesso è più facile gestire l'informazione sulle interruzioni di linea e di pagina tenendola separata dalla musica grazie a un'ulteriore voce che contenga solo pause spaziatrici e i comandi `\break`, `\pageBreak` e altre informazioni di formattazione.

Questo modello diventa utile specialmente quando si modifica `line-break-system-details` e altre utili ma lunghe proprietà di `NonMusicalPaperColumnGrob`.

```
music = \relative c'' { c4 c c c }
```

```
\score {
  \new Staff <<
    \new Voice {
      s1 * 2 \break
      s1 * 3 \break
      s1 * 6 \break
      s1 * 5 \break
    }
    \new Voice {
      \repeat unfold 2 { \music }
      \repeat unfold 3 { \music }
      \repeat unfold 6 { \music }
      \repeat unfold 5 { \music }
    }
  }
  >>
}
```



## Vedi anche

Guida alla notazione: `<undefined>` [paper variables for line breaking], pagina `<undefined>`, `<undefined>` [The layout block], pagina `<undefined>`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “LineBreakEvent” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

L’inserimento dei comandi `\autoLineBreaksOff` o `\autoBreaksOff` prima della musica produrrà dei messaggi di errore. Inserire sempre questi comandi dopo la musica.

### 4.3.2 Interruzioni di pagina

Questa sezione descrive i diversi metodi di interruzione di pagina e spiega come modificarli.

#### Interruzione di pagina manuale

L’interruzione di pagina predefinita può essere sovrascritta con i comandi `\pageBreak` o `\noPageBreak`. Questi comandi, analoghi a `\break` e `\noBreak`, devono essere inseriti dopo una stanghetta e forzano o proibiscono l’interruzione di pagina in quel punto. Ovviamente il comando `\pageBreak` forza anche un’interruzione di linea.

I comandi `\pageBreak` e `\noPageBreak` possono essere inseriti anche nel livello superiore, tra le partiture e i testi (nel blocco `markup`) di livello superiore.

In un brano, le interruzioni di pagina automatiche sono vietate per la musica compresa tra i comandi `\autoPageBreaksOff` e `\autoPageBreaksOn`. Le interruzioni di pagina manuali non sono interessate da questi comandi.

Esistono impostazioni analoghe a `ragged-right` e `ragged-last` che hanno lo stesso effetto sulla spaziatura verticale. Se `ragged-bottom` è impostato su `#t`, i sistemi non saranno giustificati verticalmente. Quando `ragged-last-bottom` è impostato su `#t` (valore predefinito), è permesso dello spazio vuoto in fondo all’ultima pagina (o in fondo all’ultima pagina di ciascun `\bookpart`). Vedi `<undefined>` [Variabili fisse della spaziatura verticale di `\paper`], pagina `<undefined>`.

Le interruzioni di pagina sono calcolate dalla funzione `page-breaking`. LilyPond fornisce tre algoritmi per calcolare le interruzioni di pagina: `ly:optimal-breaking`, `ly:page-turn-breaking` e `ly:minimal-breaking`. Quello predefinito è `ly:optimal-breaking`, ma il valore può essere modificato nel blocco `\paper`:

```
\paper {
  page-breaking = #ly:page-turn-breaking
}
```

Quando un libro ha molte partiture e pagine, il problema delle interruzioni di pagina potrebbe essere difficile da risolvere e richiedere lunghi tempi di elaborazione e molta memoria. Per semplificare il processo di interruzione delle pagine, si usano i blocchi `\bookpart` per dividere il libro in varie parti: in questo modo l'interruzione di pagina si verifica separatamente in ciascuna parte. Si possono anche usare algoritmi di interruzione di pagina diversi per le diverse parti del libro.

```
\bookpart {
  \header {
    subtitle = "Prefazione"
  }
  \paper {
    %% In una parte contenente soprattutto testo,
    %% ly:minimal-breaking potrebbe essere preferibile
    page-breaking = #ly:minimal-breaking
  }
  \markup { ... }
  ...
}
\bookpart {
  %% In questa parte, contenente musica, si usa l'algoritmo di
  %% interruzione di pagina ottimale.
  \header {
    subtitle = "Primo movimento"
  }
  \score { ... }
  ...
}
```

## Comandi predefiniti

`\pageBreak`, `\noPageBreak`, `\autoPageBreaksOn`, `\autoPageBreaksOff`.

## Vedi anche

Guida alla notazione: `<undefined>` [paper variables for page breaking], pagina `<undefined>`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

## Problemi noti e avvertimenti

Il prefisso `\once` non funziona con i comandi `\autoPageBreaksOn` e `\autoPageBreaksOff`. Se l'interruzione di pagina automatica è disabilitata e poi viene abilitata per permettere un'interruzione di pagina, deve restare attiva per alcune battute (il numero preciso di battute dipende dalla partitura) prima di essere disattivata, altrimenti la possibilità di interrompere la pagina non verrà considerata.

## Interruzione di pagina ottimale

La funzione `ly:optimal-breaking` è il metodo predefinito di LilyPond per determinare le interruzioni di pagina. Tenta di individuare un'interruzione di pagina che minimizzi la densità e l'allungamento, sia orizzontalmente che verticalmente. Diversamente da `ly:page-turn-breaking`, non prende in considerazione le voltate di pagina.

### Vedi anche

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

## Interruzione di pagina minimale

La funzione `ly:minimal-breaking` fa dei calcoli minimi per valutare l'interruzione di pagina: riempie una pagina col maggior numero possibile di sistemi prima di passare a quella successiva. Dunque potrebbe essere preferibile per le partiture con molte pagine, per le quali le altre funzioni di interruzione di pagina potrebbero essere troppo lente o richiedere troppa memoria, o con molto testo. Si abilita con:

```
\paper {
  page-breaking = #ly:minimal-breaking
}
```

### Vedi anche

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

## Interruzione di pagina su una linea

La funzione `ly:one-line-breaking` è un algoritmo di interruzione di pagina per uso speciale in quanto pone ogni brano su una sola pagina e una singola linea. Tale funzione non fa apparire titoli e margini, viene mostrato solo il brano musicale.

La larghezza della pagina sarà regolata in modo che il brano più lungo stia su una sola linea. In particolare, le variabili `paper-width`, `line-width` e `indent` del blocco `\paper` saranno ignorati, sebbene `left-margin` e `right-margin` saranno comunque considerati. L'altezza della pagina resterà invariata.

## Voltata di pagina ottimale

È spesso necessario trovare una configurazione delle interruzioni di pagina in cui ci sia una pausa al termine di ogni due pagine. In questo modo il musicista può voltare la pagina senza perdere le note. La funzione `ly:page-turn-breaking` tenta di trovare un'interruzione di pagina che minimizzi densità e allungamento, ma con l'ulteriore restrizione che le voltate di pagina sono permesse solo in punti specifici.

Ci sono due passi da seguire per usare questa funzione. Prima occorre abilitarla nel blocco `\paper`, come è spiegato in [\[Page breaking\]](#), pagina [\[undefined\]](#). Poi bisogna indicare alla funzione dove sono permesse le interruzioni di pagina.

Ci sono due modi per fare il secondo passo. Si può specificare manualmente ogni potenziale voltata di pagina, inserendo `\allowPageTurn` nei punti adatti del file di input.

Oppure, se ciò è troppo noioso, si può aggiungere l'incisore `Page_turn_engraver` a un contesto `Staff` o `Voice`. L'incisore `Page_turn_engraver` analizzerà il contesto in cerca di sezioni senza note (non cerca pause, bensì l'assenza di note, in modo che la polifonia su un singolo rigo con pause in una delle parti non confonda `Page_turn_engraver`). Quando trova una sezione senza note abbastanza lunga, `Page_turn_engraver` inserirà il comando `\allowPageTurn` nella stanghetta finale di quella sezione, a meno che non ci sia una stanghetta ‘speciale’ (come una doppia stanghetta), nel qual caso il comando `\allowPageTurn` sarà inserito nella stanghetta finale “speciale” della sezione.

L'incisore `Page_turn_engraver` legge la proprietà di contesto `minimumPageTurnLength` per determinare quanto deve essere lunga una sezione senza note prima che una voltata di pagina sia considerata. Il valore predefinito di `minimumPageTurnLength` è `(ly:make-moment 1/1)`. Per disabilitare le voltate di pagina, impostarla su un valore “molto grande”.

```
\new Staff \with { \consists "Page_turn_engraver" }
{
  a4 b c d |
  R1 | % voltata di pagina permessa qui
  a4 b c d |
  \set Staff.minimumPageTurnLength = #(ly:make-moment 5/2)
  R1 | % voltata di pagina non permessa qui
  a4 b r2 |
  R1*2 | % voltata di pagina permessa qui
  a1
}
```

In caso di ripetizioni con finali alternativi, `Page_turn_engraver` permetterà una voltata di pagina durante la ripetizione soltanto se c'è abbastanza tempo all'inizio e alla fine della ripetizione per voltare indietro la pagina. Se la ripetizione è molto breve, si può usare `Page_turn_engraver` anche per disabilitare le voltate impostando un valore per la proprietà di contesto `minimumRepeatLengthForPageTurn`, dato che `Page_turn_engraver` consente le voltate soltanto nelle ripetizioni la cui durata sia maggiore di questo valore.

I comandi per le voltate di pagina (`\pageTurn`, `\noPageTurn` e `\allowPageTurn`), possono essere usati anche nel livello superiore, nei blocchi markup di livello superiore e tra una partitura e l'altra.

## Comandi predefiniti

`\pageTurn`, `\noPageTurn`, `\allowPageTurn`.

## Vedi anche

Guida alla notazione: `<undefined>` [paper variables for line breaking], pagina `<undefined>`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

## Problemi noti e avvertimenti

Usare soltanto un incisore `Page_turn_engraver` per partitura. Se ce n'è più d'uno, interferiranno uno con l'altro.

## Vedi anche

Guida alla notazione: `<undefined>` [Vertical spacing], pagina `<undefined>`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

## 4.4 Spaziatura verticale

La spaziatura verticale è regolata da tre elementi: la quantità di spazio disponibile (ovvero il formato e i margini), la quantità di spazio tra i sistemi e la quantità di spazio tra i righi di un sistema.

### 4.4.1 Spaziatura verticale flessibile all'interno dei sistemi

Tre meccanismi distinti regolano la spaziatura verticale flessibile all'interno dei sistemi, uno per ognuna delle seguenti categorie:

- *righe non raggruppate*,

- *righi raggruppati* (righi con un gruppo come `ChoirStaff`, etc.), e
- *linee che non sono righi* (come `Lyrics`, `ChordNames`, etc.).

L'altezza di ogni sistema è determinata in due fasi. Prima vengono spaziati tutti i righi in base alla quantità di spazio disponibile. Poi le linee che non sono righi sono distribuite tra i righi.

Nota che i meccanismi di spaziatura trattati in questa sezione regolano soltanto la spaziatura verticale dei righi e delle linee (che non sono righi) all'interno di singoli sistemi. La spaziatura verticale tra sistemi, partiture, testi e margini separati è regolata dalle variabili `\paper` trattate in `<undefined>` [Variabili `\paper` della spaziatura verticale flessibile], pagina `<undefined>`.

## Proprietà della spaziatura dentro un sistema

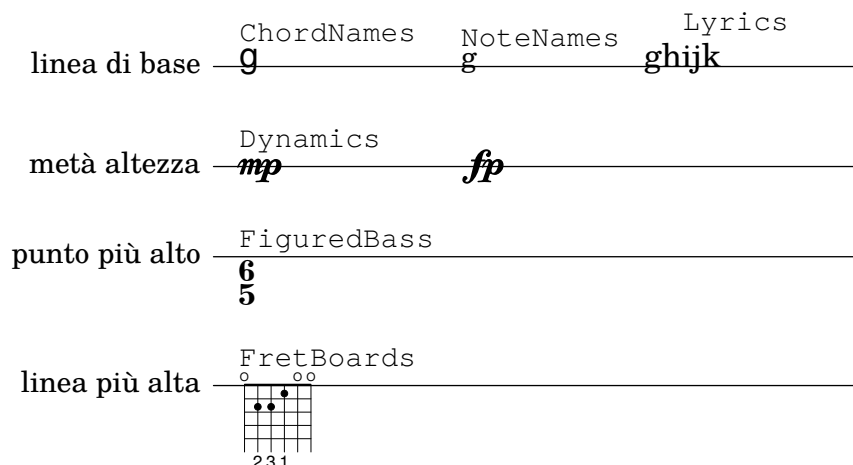
I meccanismi di spaziatura verticale dentro un sistema sono regolati da due gruppi di proprietà dei grob. Il primo gruppo è associato al grob `VerticalAxisGroup`, creato da tutti i righi e tutte le linee che non sono righi. Il secondo gruppo è associato al grob `StaffGrouper`, che può essere creato da gruppi di righi, ma solo se richiamato esplicitamente. Queste proprietà sono descritte una per una alla fine di questa sezione.

Il nome di queste proprietà (con l'eccezione di `staff-affinity`) hanno il formato `elemento1-elemento2-spacing`, dove `elemento1` e `elemento2` sono gli elementi di cui determinare la distanza. Nota che `elemento2` non è necessariamente sotto `elemento1`; per esempio, `nonstaff-relatedstaff-spacing` prenderà le misure verso l'alto a partire dalla linea che non è un rigo (`nonstaff`) se `staff-affinity` è impostato su UP.

Ogni distanza è calcolata tra i *punti di riferimento* dei due elementi. Il punto di riferimento di un rigo è il centro verticale del suo `StaffSymbol` (ovvero la linea centrale se `line-count` (il numero di linee) è dispari; lo spazio centrale se `line-count` è pari). I punti di riferimento per ciascuna linea che non è un rigo sono elencati nella seguente tabella:

Linea non-rigo	Punto di riferimento
<code>ChordNames</code>	linea di base
<code>NoteNames</code>	linea di base
<code>Lyrics</code>	linea di base
<code>Dynamics</code>	metà altezza di 'm'
<code>FiguredBass</code>	punto più alto
<code>FretBoards</code>	linea più alta

Nell'immagine seguente, le linee orizzontali indicano le posizioni di questi punti di riferimento:



Tutte le proprietà di spaziatura verticale del grob (eccetto `staff-affinity`) usano la stessa struttura della lista associativa usata dalle variabili di spaziatura di `\paper` trattate in [\(undefined\)](#) [Variabili `\paper` della spaziatura verticale flessibile], pagina [\(undefined\)](#). Metodi specifici per modificare queste liste sono spiegati in Sezione 5.3.6 [Modifying alists], pagina 608. Le proprietà dei grob devono essere modificate con un comando `\override` dentro un blocco `\score` o `\layout` e non in un blocco `\paper`.

L'esempio seguente illustra i due modi con cui si possono modificare queste liste associative (alist). La prima dichiarazione trasforma un elemento-valore singolarmente, mentre la seconda ridefinisce completamente la proprietà:

```
\new Staff \with {
  \override VerticalAxisGroup.default-staff-staff-spacing.basic-distance = #10
} { ... }

\new Staff \with {
  \override VerticalAxisGroup.default-staff-staff-spacing =
    #'((basic-distance . 10)
      (minimum-distance . 9)
      (padding . 1)
      (stretchability . 10))
} { ... }
```

Per cambiare le impostazioni di spaziatura globalmente, inserirle in un blocco `\layout`:

```
\layout {
  \context {
    \Staff
    \override VerticalAxisGroup.default-staff-staff-spacing.basic-distance = #10
  }
}
```

Le impostazioni predefinite delle proprietà di spaziatura verticale dei grob sono elencate in Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno* e Sezione “StaffGrouper” in *Guida al Funzionamento Interno*. Le modifiche predefinite con `\override` per tipologie specifiche di linee che non sono righi sono elencate nelle descrizioni del relativo contesto in Sezione “Contexts” in *Guida al Funzionamento Interno*.

## Proprietà del grob VerticalAxisGroup

Le proprietà di `VerticalAxisGroup` sono solitamente modificate con un `\override` nel livello `Staff` (o equivalente).

### `staff-staff-spacing`

Usata per determinare la distanza tra il rigo corrente e il rigo inferiore nello stesso sistema, anche se tra i due si trovano una o più linee che non sono righi (come *Lyrics*). Non è applicata all'ultimo rigo di un sistema.

Inizialmente, la proprietà `staff-staff-spacing` di un `VerticalAxisGroup` è una funzione Scheme che applica le proprietà di `StaffGrouper` se il rigo fa parte di un gruppo, o la proprietà `default-staff-staff-spacing` del rigo altrimenti. Questo permette ai rigi di essere spaziati diversamente quando sono raggruppati. Per ottenere una spaziatura uniforme indipendentemente dal raggruppamento, questa funzione può essere sostituita da un alist di spaziatura flessibile, usando la forma di `override` che ridefinisce completamente la variabile, come mostrato prima.



**default-staff-staff-spacing**

Un alist di spaziatura flessibile che definisce la proprietà **staff-staff-spacing** usata per i righi isolati, a meno che **staff-staff-spacing** non sia stata impostata esplicitamente con un `\override`.

**staff-affinity**

La direzione del rigo da usare per spaziare la linea che non è un rigo. Le opzioni sono UP (su), DOWN (giù) e CENTER (centro). Se impostata su CENTER, la linea fuori dal rigo si troverà in un punto equidistante tra i due righi più vicini su qualunque lato, a meno che delle collisioni o altre costrizioni di spazio non lo impediscano. Linee (che non sono righi) adiacenti dovrebbero avere un valore di **staff-affinity** che non cresce: per esempio, una linea che non è un rigo impostata su UP non deve seguire immediatamente una linea impostata su DOWN. Linee che non sono righi in cima a un sistema devono usare DOWN; quelle in fondo UP. Impostando **staff-affinity** per un rigo, questo sarà trattato come una linea che non è un rigo. Impostando **staff-affinity** su #f, una linea che non è un rigo sarà trattata come un rigo. Impostando **staff-affinity** su UP, CENTER o DOWN, un rigo verrà spaziato come se fosse una linea che non è un rigo.

**nonstaff-relatedstaff-spacing**

La distanza fra la linea (che non è un rigo) corrente e il rigo più vicino nella direzione di **staff-affinity**, se non ci sono linee che non sono righi tra le due e **staff-affinity** è impostato su UP o DOWN. Se **staff-affinity** è impostato su CENTER, viene usato **nonstaff-relatedstaff-spacing** per i righi più vicini su *entrambi* i lati, anche se appaiono altre linee tra quella corrente e uno qualsiasi dei righi. Ciò significa che il posizionamento di una linea dipende sia dai righi che dalle linee circostanti. Impostando la proprietà **stretchability** di uno di questi tipi di spaziatura su un piccolo valore, quella spaziatura sarà dominante. Impostando **stretchability** su un grande valore, quella spaziatura avrà poco effetto.

**nonstaff-nonstaff-spacing**

La distanza fra la linea (che non è un rigo) corrente e quella successiva nella direzione di **staff-affinity**, se entrambe sono sullo stesso lato del rigo in questione e se **staff-affinity** è impostata su UP o DOWN.

**nonstaff-unrelatedstaff-spacing**

La distanza fra la linea (che non è un rigo) corrente e il rigo nella direzione opposta rispetto a **staff-affinity**, se non ci sono altre linee tra i due e se **staff-affinity** è impostato su UP o DOWN. Ciò può servire, per esempio, a imporre un padding minimo tra una linea Lyrics e il rigo al quale non appartiene.

## Proprietà del grob StaffGrouper

Le proprietà di **StaffGrouper** sono solitamente modificate con un `\override` nel livello **StaffGroup** (o livello equivalente).

**staff-staff-spacing**

La distanza tra righi consecutivi del gruppo di righi corrente. La proprietà **staff-staff-spacing** del grob **VerticalAxisGroup** di un singolo rigo può essere sovrascritta con varie impostazioni di spaziatura per quel rigo.

**staffgroup-staff-spacing**

La distanza tra l'ultimo rigo del gruppo di righi corrente e il rigo immediatamente successivo nello stesso sistema, anche se tra i due righi ci sono una o più linee che non sono righi (come Lyrics). Non è applicata al rigo inferiore di un sistema. La proprietà **staff-staff-spacing** del grob **VerticalAxisGroup** di un singolo rigo può essere sovrascritta con varie impostazioni di spaziatura per quel rigo.

## Vedi anche

Guida alla notazione: `<undefined>` [Variabili `\paper` della spaziatura verticale flessibile], pagina `<undefined>`, Sezione 5.3.6 [Modifying alists], pagina 608.

File installati: `ly/engraver-init.ly`, `scm/define-grobs.scm`.

Guida al funzionamento interno: Sezione “Contexts” in *Guida al Funzionamento Interno*, Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*, Sezione “StaffGrouper” in *Guida al Funzionamento Interno*.

## Spaziatura dei righi non raggruppati

I *righe* (come `Staff`, `DrumStaff`, `TabStaff`, etc.) sono contesti che possono contenere uno o più contesti voce, ma non possono contenere altri righe.

Le seguenti proprietà influenzano la spaziatura di righe *non raggruppati*:

- Proprietà di `VerticalAxisGroup`:
  - `default-staff-staff-spacing`
  - `staff-staff-spacing`

Queste proprietà del grob sono state descritte una a una in precedenza; vedi `<undefined>` [Within-system spacing properties], pagina `<undefined>`.

Altre proprietà entrano in gioco per i righe che sono parte di un gruppo; vedi `<undefined>` [Spacing of grouped staves], pagina `<undefined>`.

L'esempio seguente mostra come la proprietà `default-staff-staff-spacing` possa influenzare la spaziatura di righe non raggruppati. Le stesse modifiche applicate a `staff-staff-spacing` avrebbero lo stesso effetto, ma verrebbero applicate anche nel caso in cui i righe siano combinati in uno o più gruppi.

```
\layout {
  \context {
    \Staff
    \override VerticalAxisGroup.default-staff-staff-spacing =
      #'((basic-distance . 8)
        (minimum-distance . 7)
        (padding . 1))
  }
}

<<
% Questa nota molto bassa ha bisogno di più spazio di quanto 'basic-distance
% possa fornirne, dunque la distanza tra questo rigo e quello successivo
% è determinato da 'padding.
\new Staff { b,2 r | }

% Qui 'basic-distance fornisce abbastanza spazio, e non c'è bisogno
% di comprimere lo spazio (verso 'minimum-distance) per far spazio
% per qualcos'altro sulla pagina, dunque la distanza tra questo
% rigo e quello successivo è determinato da 'basic-distance.
\new Staff { \clef bass g2 r | }

% Impostando 'padding su un valore negativo, è possibile far sì che
% i righe entrino in collisione. Il più basso valore accettabile per
% 'basic-distance è 0.
\new Staff \with {
```

```

\override VerticalAxisGroup.default-staff-staff-spacing =
  #'((basic-distance . 3.5)
      (padding . -10))
} { \clef bass g2 r | }
\new Staff { \clef bass g2 r | }
>>

```



## Vedi anche

File installati: `scm/define-grobs.scm`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*.

## Spaziatura dei rigli raggruppati

Nelle partiture orchestrali e in alte grosse partiture, di norma i rigli vengono raggruppati. Lo spazio tra i gruppi è più ampio dello spazio tra i rigli dello stesso gruppo.

I *gruppi di rigli* (come `StaffGroup`, `ChoirStaff`, etc.) sono contesti che possono contenere uno o più rigli simultaneamente.

Le seguenti proprietà influenzano la spaziatura dei rigli nei gruppi:

- Proprietà di `VerticalAxisGroup`:
  - `staff-staff-spacing`
- Proprietà di `StaffGrouper`:
  - `staff-staff-spacing`
  - `staffgroup-staff-spacing`

Queste proprietà dei grob sono descritte una a una in una sezione precedente; vedi `<undefined>` [Within-system spacing properties], pagina `<undefined>`.

L'esempio seguente mostra come le proprietà del grob `StaffGrouper` possano influenzare la spaziatura dei rigli raggruppati:

```

\layout {
  \context {
    \Score
    \override StaffGrouper.staff-staff-spacing.padding = #0
    \override StaffGrouper.staff-staff-spacing.basic-distance = #1
  }
}

<<
\new PianoStaff \with {

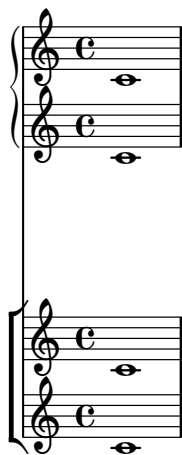
```

```

\override StaffGrouper.staffgroup-staff-spacing.basic-distance = #20
} <<
\new Staff { c'1 }
\new Staff { c'1 }
>>

\new StaffGroup <<
\new Staff { c'1 }
\new Staff { c'1 }
>>
>>

```



## Vedi anche

File installati: `scm/define-grobs.scm`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*, Sezione “StaffGrouper” in *Guida al Funzionamento Interno*.

## Spaziatura delle linee che non sono righi

Le *linee che non sono righi* (come `Lyrics`, `ChordNames`, etc.) sono contesti i cui oggetti della formattazione sono disposti come se fossero su dei righi (ovvero su linee orizzontali all’interno dei sistemi). Precisamente, le linee che non sono righi sono contesti non-rigo che contengono l’incisore Sezione “`Axis_group_engraver`” in *Guida al Funzionamento Interno*.

Le seguenti proprietà influenzano la spaziatura delle linee che non sono righi:

- Proprietà di `VerticalAxisGroup`:
  - `staff-affinity`
  - `nonstaff-relatedstaff-spacing`
  - `nonstaff-nonstaff-spacing`
  - `nonstaff-unrelatedstaff-spacing`

Queste proprietà del grob sono descritte una a una in una sezione precedente, vedi `<undefined>` [Within-system spacing properties], pagina `<undefined>`.

L’esempio seguente mostra come la proprietà `nonstaff-nonstaff-spacing` influenza la spaziatura di linee che non sono un rigo consecutive. Impostando l’elemento `stretchability` su un valore molto alto, il testo vocale riesce a allungarsi molto più del solito:

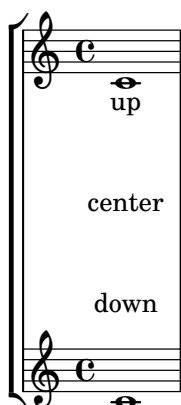
```
\layout {
```

```

\context {
  \Lyrics
  \override VerticalAxisGroup.nonstaff-nonstaff-spacing.stretchability = #1000
}
}

\new StaffGroup
<<
  \new Staff \with {
    \override VerticalAxisGroup.staff-staff-spacing = #'((basic-distance . 30))
  } { c'1 }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #UP
  } \lyricmode { up }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #CENTER
  } \lyricmode { center }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #DOWN
  } \lyricmode { down }
  \new Staff { c'1 }
>>

```



## Vedi anche

File installati: `ly/engraver-init.ly`, `scm/define-grobs.scm`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Contexts” in *Guida al Funzionamento Interno*, Sezione “VerticalAxisGroup” in *Guida al Funzionamento Interno*.

### 4.4.2 Posizionamento esplicito di righe e sistemi

Un modo per comprendere i meccanismi di spaziatura verticale appena spiegati è di considerarli come un insieme di impostazioni che regolano la quantità di *padding* verticale tra righe e tra sistemi.

È possibile gestire la spaziatura verticale in un modo diverso usando `NonMusicalPaperColumn.line-break-system-details`. Mentre i meccanismi di spaziatura verticale flessibile specificano il padding verticale, `NonMusicalPaperColumn.line-break-system-details` indica precisamente le posizioni verticali esatte sulla pagina.

`NonMusicalPaperColumn.line-break-system-details` accetta una lista associativa di tre diverse impostazioni:

- `X-offset`
- `Y-offset`
- `alignment-distances`

Le modifiche del `grob` con `\override`, incluse quelle per `NonMusicalPaperColumn` come nell'esempio successivo, possono trovarsi in uno di questi tre diversi punti del file di input:

- direttamente in mezzo alle note
- in un blocco `\context`
- nel blocco `\with`

Quando si modifica `NonMusicalPaperColumn`, si usa il solito comando `\override` nei blocchi `\context` e nel blocco `\with`. Invece quando si modifica `NonMusicalPaperColumn` in mezzo alle note, si usa il comando speciale `\overrideProperty`. Ecco alcuni esempi di modifiche di `NonMusicalPaperColumn` col comando speciale `\overrideProperty`:

```
\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((X-offset . 20))
```

```
\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((Y-offset . 40))
```

```
\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((X-offset . 20)
    (Y-offset . 40))
```

```
\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((alignment-distances . (15)))
```

```
\overrideProperty NonMusicalPaperColumn.line-break-system-details
  #'((X-offset . 20)
    (Y-offset . 40)
    (alignment-distances . (15)))
```

Per comprendere come funziona ognuna di queste impostazioni, iniziamo vedendo un esempio che non contiene alcuna modifica.

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          s1*5 \break
          s1*5 \break
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
    \new Staff {
      \repeat unfold 15 { d'4 d' d' d' }
    }
  }
}
```



Questa partitura isola l'informazione sulle interruzioni di linea e di pagina in una voce apposita. Questa tecnica di creare una voce per le interruzioni permette di tenere la formattazione separata dalla musica via via che il nostro esempio diventa più complicato. Vedi `<undefined>` [Breaks], pagina `<undefined>`.

I `\break` espliciti dividono proporzionalmente la musica in cinque misure per linea. La spaziatura verticale è quella predefinita di LilyPond. Per impostare esplicitamente il punto di inizio verticale di ogni sistema, possiamo impostare la coppia `Y-offset` dell'attributo `line-break-system-details` del grob `NonMusicalPaperColumn`:

```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 0))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 40))
          s1*5 \break
          \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
            #'((Y-offset . 60))
          s1*5 \break
        }
        \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
      >>
    \new Staff {
      \repeat unfold 15 { d'4 d' d' d' }
```

```

    }
  >>
}
}

```



Nota che `line-break-system-details` accetta una lista associativa di molti valori, ma ne abbiamo impostato solo uno in questo esempio. Nota anche che la proprietà `Y-offset` qui determina la posizione verticale esatta sulla pagina in cui ogni nuovo sistema verrà visualizzato.

Ora che abbiamo impostato esplicitamente il punto di inizio verticale di ogni sistema, possiamo impostare manualmente anche le distanze verticali tra i righi. Per farlo usiamo la sottoproprietà `alignment-distances` di `line-break-system-details`.

```

\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
    \new Staff <<
    \new Voice {
      \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
        #'((Y-offset . 20)
          (alignment-distances . (10)))
      s1*5 \break
      \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
        #'((Y-offset . 60)
          (alignment-distances . (15)))
      s1*5 \break
      \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details

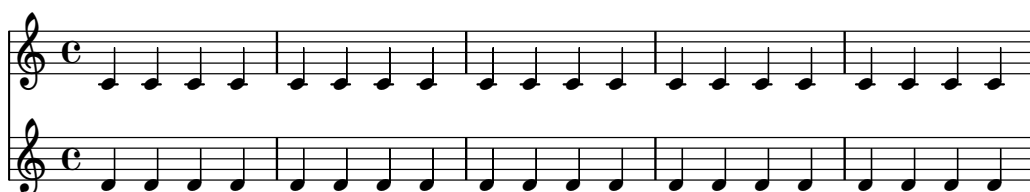
```



```

      #'((Y-offset . 85)
        (alignment-distances . (20)))
      s1*5 \break
    }
    \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
  >>
  \new Staff {
    \repeat unfold 15 { d'4 d' d' d' }
  }
  >>
}
}

```



Nota che qui assegnamo due valori diversi all'attributo `line-break-system-details` del grob `NonMusicalPaperColumn`. Sebbene l'attributo `alist line-break-system-details` accetti molti altri parametri di spaziatura (inclusa, per esempio, una coppia corrispondente di `X-offset`), è sufficiente impostare soltanto le coppie `Y-offset` e `alignment-distances` per regolare il punto di inizio verticale di ogni sistema e ogni rigo. Infine nota che `alignment-distances` specifica il posizionamento verticale dei rigi ma non dei gruppi di rigi.

```

\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {

```

```

\score {
  <<
    \new Staff <<
      \new Voice {
        \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
          #'((Y-offset . 0)
            (alignment-distances . (30 10)))
        s1*5 \break
        \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
          #'((Y-offset . 60)
            (alignment-distances . (10 10)))
        s1*5 \break
        \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
          #'((Y-offset . 100)
            (alignment-distances . (10 30)))
        s1*5 \break
      }
      \new Voice { \repeat unfold 15 { c'4 c' c' c' } }
    >>
    \new StaffGroup <<
      \new Staff { \repeat unfold 15 { d'4 d' d' d' } }
      \new Staff { \repeat unfold 15 { e'4 e' e' e' } }
    >>
  >>
}

```

The image displays three systems of musical notation, each consisting of a single treble staff and a grand staff (treble and bass). The first system shows a single melodic line in the treble staff and a continuous bass line. The second system shows a single melodic line in the treble staff and a continuous bass line. The third system shows a single melodic line in the treble staff and a continuous bass line.

Alcuni punti da considerare:

- Quando si usa `alignment-distances`, il testo vocale e altre linee che non sono righe non contano come rigo.
- Le unità dei numeri assegnati a `X-offset`, `Y-offset` e `alignment-distances` sono interpretati come multipli della distanza tra linee del rigo adiacenti. Valori positivi spostano in su righe e testo, valori negativi li spostano in giù.
- Dato che le impostazioni di `NonMusicalPaperColumn.line-break-system-details` illustrano qui permettono il posizionamento di righe e sistemi ovunque sulla pagina, è possibile

violare i confini del foglio o dei margini o perfino sovrapporre righe e sistemi uno sopra l'altro. Ciò può essere evitato assegnando valori ragionevoli a queste diverse impostazioni.

## Vedi anche

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

### 4.4.3 Elusione delle collisioni verticali

Intuitivamente, ci sono alcuni oggetti della notazione musicale che appartengono al rigo e altri che devono essere disposti fuori dal rigo. Gli oggetti esterni al rigo comprendono i numeri di chiamata, il testo e le dinamiche (d'ora in avanti tutti questi elementi saranno chiamati oggetti esterni al rigo). La regola di LilyPond per il posizionamento verticale degli oggetti esterni al rigo è di disporli il più vicino possibile al rigo ma non così vicino da farli collidere con un altro oggetto.

LilyPond usa la proprietà `outside-staff-priority` per determinare se un grob è un oggetto fuori dal rigo: se `outside-staff-priority` è un numero, il grob è un oggetto esterno al rigo. `outside-staff-priority` indica a LilyPond anche in quale ordine disporre gli oggetti.

LilyPond posiziona prima tutti gli oggetti che non sono esterni al rigo. Quindi ordina gli oggetti esterni al rigo in base al loro valore di `outside-staff-priority` (in ordine crescente). Uno per volta, LilyPond prende gli oggetti esterni al rigo e li dispone in modo che non entrino in collisione con alcun oggetto che sia già stato disposto. Ovvero, se due grob esterni al rigo si contendono lo stesso spazio, quello col valore di `outside-staff-priority` più basso sarà posto più vicino al rigo.

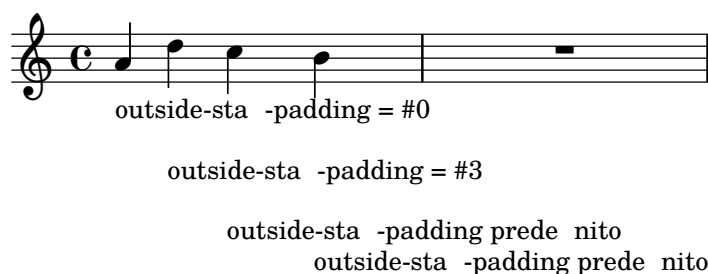
```
\relative c' ' {
  c4_"Testo"\pp
  r2.
  \once \override TextScript.outside-staff-priority = #1
  c4_"Testo"\pp % stavolta il testo sarà più vicino al rigo
  r2.
  % impostando outside-staff-priority su un non-numero,
  % disabilitiamo l'elusione automatica delle collisioni
  \once \override TextScript.outside-staff-priority = ##f
  \once \override DynamicLineSpanner.outside-staff-priority = ##f
  c4_"Testo"\pp % qui entrano in collisione
}
```



Il padding verticale intorno agli oggetti esterni al rigo può essere regolato con `outside-staff-padding`.

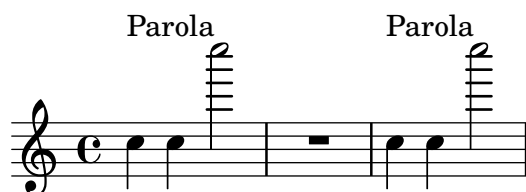
```
\relative {
  \once \override TextScript.outside-staff-padding = #0
  a'4-"outside-staff-padding = #0"
  \once \override TextScript.outside-staff-padding = #3
  d-"outside-staff-padding = #3"
  c-"outside-staff-padding predefinito"
  b-"outside-staff-padding predefinito"
  R1
}
```

}



Per impostazione predefinita, gli oggetti esterni al rigo sono disposti in modo da evitare la collisione orizzontale con grob posizionati precedentemente. Ciò può portare a situazioni in cui gli oggetti sono posizionati uno vicino all'altro orizzontalmente. Come è dimostrato nell'esempio successivo, impostando `outside-staff-horizontal-padding` si aumenta la spaziatura orizzontale richiesta e in questo caso si sposta in su il testo per impedire che si avvicini troppo ai tagli addizionali.

```
\relative {
  c''4^"Parola" c c''2
  R1
  \once \override TextScript.outside-staff-horizontal-padding = #1
  c,,4^"Parola" c c''2
}
```



## Vedi anche

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

## 4.5 Spaziatura orizzontale

### 4.5.1 Panoramica sulla spaziatura orizzontale

Il motore della spaziatura traduce le differenze delle durate delle note in distanze allungabili (‘springs’) di diversa lunghezza. Durate più lunghe occupano più spazio, quelle più brevi ne occupano meno. Le durate più brevi occupano una quantità fissa di spazio (regolata da `shortest-duration-space` nell’oggetto Sezione “SpacingSpanner” in *Guida al Funzionamento Interno*). Più lunga è la durata, più spazio occupa: raddoppiando una durata si aggiunge spazio alla nota di una quantità pari al valore di `spacing-increment`.

Per esempio, il brano seguente contiene molte minime, semiminime e crome; la croma (1/8) è seguita da 1 Larghezza della Testa di Nota (LTN). La semiminima (1/4) è seguita da 2 LTN, la minima (1/2) da 3 LTN, etc.

```
\relative c' {
  c2 c4. c8
  c4. c8 c4. c8
  c8 c c4 c c
}
```



Solitamente, `spacing-increment` è impostato su 1.2 di spazio rigo, che equivale all'incirca alla larghezza della testa di nota, e `shortest-duration-space` è impostato su 2.0, che significa che la nota più breve occupa 2.4 di spazio rigo (2 volte `spacing-increment`) di spazio orizzontale. Questo spazio è calcolato dal margine sinistro del simbolo, dunque le note più brevi sono generalmente seguite da un LTN di spazio.

Se si seguisse esattamente la procedura descritta, aggiungendo una sola biscroma (1/32) a un brano che usa solo crome e semicrome, la spaziatura orizzontale dell'intero brano sarebbe troppo larga. Infatti la nota più breve non è più una semicroma ma una biscroma, aggiungendo quindi 1 LTN a ogni nota. Per impedire ciò, la durata più breve per la spaziatura non è la nota più breve, bensì la che ricorre più frequentemente nel brano.

La durata più breve più comune viene individuata nel modo seguente. In ogni misura viene determinata la durata più breve e quella più frequente viene scelta come base per la spaziatura, con la condizione che tale durata debba essere sempre uguale o inferiore a una nota di un ottavo.

Tali durate possono anche essere personalizzate. Impostando `common-shortest-duration` in Sezione “SpacingSpanner” in *Guida al Funzionamento Interno*, si imposta la durata di base per la spaziatura. La durata massima per essa (solitamente un ottavo), si imposta con `base-shortest-duration`.

Note ancora più brevi della nota più breve più comune sono seguite da uno spazio proporzionale alla loro durata rispetto a essa. Dunque se aggiungessimo solo alcuni sedicesimi all'esempio precedente, sarebbero seguiti dalla metà di LTN:

```
\relative { c''2 c4. c8 | c4. c16[ c] c4. c8 | c8 c c4 c c }
```



Nel saggio *Essay on automated music engraving*, è spiegato che le direzioni del gambo influenzano la spaziatura (vedi Sezione “Optical spacing” in *Saggio*). Ciò è regolato dalla proprietà `stem-spacing-correction` nell'oggetto Sezione “NoteSpacing” in *Guida al Funzionamento Interno*. Questi oggetti sono generati per ogni contesto Sezione “Voice” in *Guida al Funzionamento Interno*. L'oggetto `StaffSpacing` (generato nel contesto Sezione “Staff” in *Guida al Funzionamento Interno*) contiene la stessa proprietà per regolare la spaziatura tra gambo e stanghetta. L'esempio seguente mostra queste correzioni, una volta con le impostazioni predefinite e una volta con correzioni esagerate:



LilyPond supporta la notazione proporzionale, vedi [\[Proportional notation\]](#), pagina [\[undefined\]](#).

## Vedi anche

Essay on automated music engraving: Sezione “Optical spacing” in *Saggio*.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “SpacingSpanner” in *Guida al Funzionamento Interno*, Sezione “NoteSpacing” in *Guida al Funzionamento Interno*, Sezione “StaffSpacing” in *Guida al Funzionamento Interno*, Sezione “NonMusicalPaperColumn” in *Guida al Funzionamento Interno*.

## Problemi noti e avvertimenti

Non esiste un modo semplice per modificare manualmente la spaziatura. Per aggirare il problema si può inserire dell'ulteriore spazio in una partitura, regolando il valore di padding di quanto è necessario:

```
\override Score.NonMusicalPaperColumn.padding = #10
```

Non esiste alcun trucco per diminuire la quantità di spazio.

### 4.5.2 Nuova spaziatura nel corso di un brano

Nuove sezioni con parametri di spaziatura diversi possono essere iniziati con `newSpacingSection`. Ciò può essere utile quando ci sono sezioni con nozioni diverse di note lunghe e brevi.

Nell'esempio seguente, il cambio di indicazione di tempo introduce una nuova sezione, quindi i sedicesimi hanno automaticamente una spaziatura un po' più larga.

```
\relative c' {
  \time 2/4
  c4 c8 c
  c8 c c4 c16[ c c8] c4
  \newSpacingSection
  \time 4/16
  c16[ c c8]
}
```



Il comando `\newSpacingSection` crea un nuovo oggetto `SpacingSpanner` in quel momento musicale. Se gli aggiustamenti automatici della spaziatura non producono la spaziatura richiesta, si possono applicare degli `\override` manuali alle sue proprietà. Questi devono essere applicati nello stesso momento musicale del comando `\newSpacingSection` stesso. Avranno effetto sulla spaziatura di tutta la musica seguente finché le proprietà non vengono cambiate in una nuova sezione. Per esempio:

```
\relative c' {
  \time 4/16
  c16[ c c8]
  \newSpacingSection
  \override Score.SpacingSpanner.spacing-increment = #2
  c16[ c c8]
  \newSpacingSection
  \revert Score.SpacingSpanner.spacing-increment
  c16[ c c8]
}
```



## Vedi anche

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “SpacingSpanner” in *Guida al Funzionamento Interno*.

### 4.5.3 Modifica della spaziatura orizzontale

La spaziatura orizzontale può essere modificata tramite la proprietà `base-shortest-duration`. Nel prossimo esempio confrontiamo la stessa musica, prima senza cambiare la proprietà e poi cambiandola. Valori più grandi di `ly:make-moment` produrranno musica più densa. Nota che `ly:make-moment` costituisce una durata, dunque 1 4 è una durata più lunga di 1 16.

```
\score {
  \relative {
    g'4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  }
}
```



```
\score {
  \relative {
    g'4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner.base-shortest-duration = #(ly:make-moment 1/16)
    }
  }
}
```



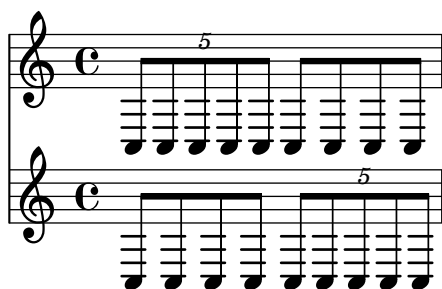




### Frammenti di codice selezionati

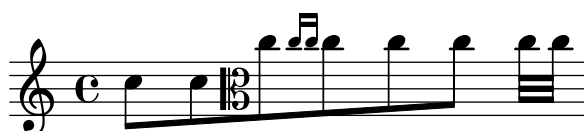
L'impostazione predefinita prevede che la spaziatura nei gruppi irregolari dipenda da vari fattori diversi dalla durata (come alterazioni, cambi di chiave, etc). Per ignorare tali simboli e forzare la spaziatura perché sia uniforme, usare `Score.SpacingSpanner.uniform-stretching`. Questa proprietà può essere modificata soltanto all'inizio di una partitura:

```
\score {
  <<
    \new Staff {
      \tuplet 5/4 { c8 c c c c } c8 c c c
    }
    \new Staff {
      c8 c c c \tuplet 5/4 { c8 c c c c }
    }
  >>
  \layout {
    \context {
      \Score
      \override SpacingSpanner.uniform-stretching = ##t
    }
  }
}
```



Se si imposta `strict-note-spacing`, la spaziatura tra le note non tiene conto di chiavi, stanghette e abbellimenti:

```
\override Score.SpacingSpanner.strict-note-spacing = ##t
\new Staff \relative {
  c''8[ c \clef alto c \grace { c16 c } c8 c c] c32[ c] }
```



## Vedi anche

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

### 4.5.4 Larghezza della linea

Le impostazioni fondamentali che influenzano la spaziatura sono `indent` e `line-width`, impostate nel blocco `\layout`. Regolano l’indentazione della prima linea musicale e la lunghezza delle linee.

Se `ragged-right` è impostato su vero nel blocco `\layout`, i sistemi terminano alla loro naturale lunghezza orizzontale, invece di essere espansi orizzontalmente per riempire tutta la linea. Ciò è utile in caso di brevi frammenti e per verificare quanto è compatta la spaziatura naturale. L’impostazione predefinita è solitamente falso, ma se la partitura ha un solo sistema il valore predefinito è vero.

L’opzione `ragged-last` è simile a `ragged-right`, ma ha effetto soltanto sull’ultima linea del brano. Nessune restrizioni vengono poste su quella linea. Il risultato è simile alla formattazione dei paragrafi di testo. In un paragrafo l’ultima linea occupa la sua naturale lunghezza orizzontale.

```
\layout {
  indent = #0
  line-width = #150
  ragged-last = ##t
}
```

## Vedi anche

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

### 4.5.5 Notazione proporzionale

LilyPond supporta la notazione proporzionale, un tipo di spaziatura orizzontale in cui ogni nota occupa una quantità di spazio orizzontale esattamente equivalente alla sua durata musicale. Questo tipo di spaziatura proporzionale può essere paragonata alla spaziatura orizzontale su carta quadrettata. Alcune partiture della fine del ventesimo secolo e dell’inizio del ventunesimo usano la notazione proporzionale per chiarire relazioni ritmiche complesse o per agevolare il posizionamento della linea del tempo o di altri elementi grafici direttamente nella partitura.

LilyPond supporta cinque diverse impostazioni per la notazione proporzionale, che possono essere usate insieme o da sole:

- `proportionalNotationDuration`
- `uniform-stretching`
- `strict-note-spacing`
- `\remove "Separating_line_group_engraver"`
- `\override PaperColumn.used = ##t`

Nell’esempio seguente analizziamo queste cinque diverse impostazioni di notazione proporzionale e valutiamo come esse interagiscono tra loro.

Iniziamo con l’esempio seguente di una misura, che usa la spaziatura classica con la giustificazione del rigo disattivata.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
}
```

}



La minima all’inizio della misura occupa uno spazio molto inferiore alla metà dello spazio orizzontale della misura. Ugualmente, i sedicesimi e le quintine di sedicesimi alla fine della battuta insieme occupano molto più spazio della metà dello spazio orizzontale della misura.

Nell’incisione tipografica classica, questa spaziatura è solitamente proprio ciò che si desidera, perché è possibile prendere in prestito dello spazio orizzontale dalla minima e economizzare lo spazio orizzontale complessivo della misura.

D’altra parte, se vogliamo inserire una linea del tempo con tacche o altri elementi grafici sopra o sotto la partitura, abbiamo bisogno della notazione proporzionale. Per attivarla si usa l’impostazione `proportionalNotationDuration`.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/20)
    }
  }
}
```



La minima all’inizio della misura e le note più veloci nella seconda metà della misura ora occupano la stessa quantità di spazio orizzontale. Potremmo inserire una linea del tempo con tacche o un’immagine grafica sopra o sotto questo esempio.

L’impostazione `proportionalNotationDuration` è un’impostazione di contesto che si trova in `Score`. Ricordiamo che le impostazioni di contesto possono apparire in tre luoghi del file di input: in un blocco `\with`, in un blocco `\context` o direttamente in mezzo alle note preceduta dal comando `\set`. Come per tutte le impostazioni di contesto, l’utente può scegliere in quale di questi tre luoghi impostare `proportionalNotationDuration`.

L’impostazione `proportionalNotationDuration` prende un solo argomento, che è la durata di riferimento in base alla quale tutta la musica verrà spaziata. La funzione Scheme di LilyPond `make-moment` prende due argomenti: un numeratore e un denominatore che insieme esprimono una qualche frazione di una nota intera. La funzione di chiamata `(ly:make-moment 1/20)` produce quindi una durata di riferimento di un ventesimo di nota. Sono ammessi anche valori come `(ly:make-moment 1/16)`, `(ly:make-moment 1/8)` e `(ly:make-moment 3/97)`.

Come scegliamo la durata di riferimento corretta da passare a `proportionalNotationDuration`? Solitamente attraverso un processo di prova e errore, iniziando con una durata vicina alla più veloce (o piccola) durata del brano. Durate di riferimento più piccole determinano una spaziatura della musica più larga; quelle più grandi causano una spaziatura più stretta.

```
\score {
```

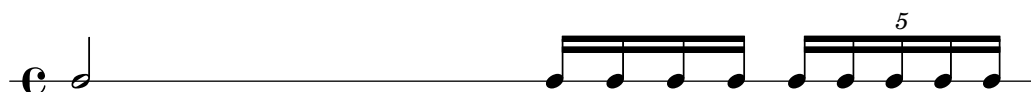
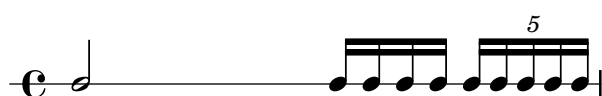
```

<<
  \new RhythmicStaff {
    c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
  }
>>
\layout {
  \context {
    \Score
    proportionalNotationDuration = #(ly:make-moment 1/8)
  }
}

\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/16)
    }
  }
}

\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/32)
    }
  }
}

```

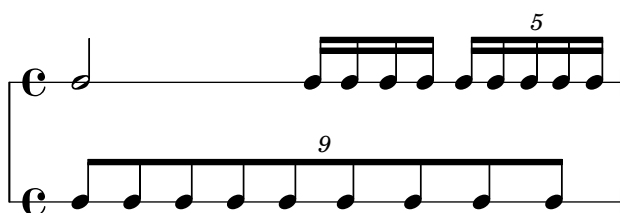


Nota che una durata di riferimento troppo grande – come la nota di un ottavo, sopra – determina una spaziatura della musica troppo stretta e può causare collisioni tra le teste di nota. Fare attenzione anche al fatto che la notazione proporzionale in generale occupa più spazio orizzontale della spaziatura classica. Insomma, la spaziatura proporzionale fornisce chiarezza ritmica al costo dello spazio orizzontale.

Ora vediamo come spaziare in modo ottimale i gruppi irregolari sovrapposti.

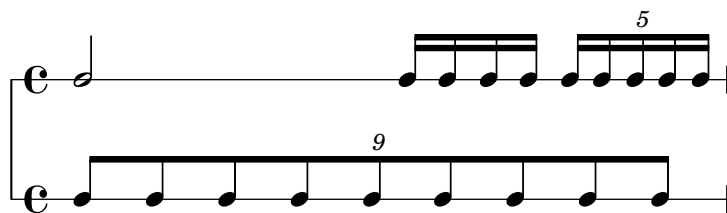
Iniziamo esaminando cosa succede al nostro esempio di partenza, con la spaziatura classica, quando aggiungiamo un secondo rigo con un diverso tipo di gruppo irregolare.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
    \new RhythmicStaff {
      \tuplet 9/8 { c8 8 8 8 8 8 8 8 8 }
    }
  >>
}
```



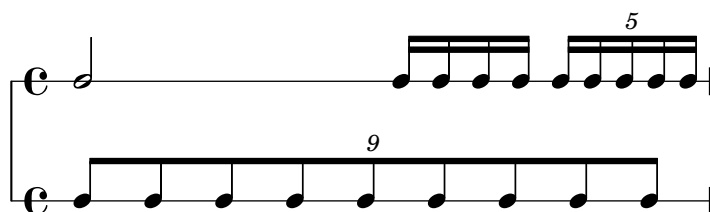
La spaziatura è pessima perché le note del rigo inferiore spaziate a distanze uguali non si allungano in modo uniforme. Le incisioni classiche contengono pochissime terzine complesse e quindi le regole di incisione classica possono generare questo tipo di risultato. Impostando `proportionalNotationDuration` ciò viene corretto.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
    \new RhythmicStaff {
      \tuplet 9/8 { c8 8 8 8 8 8 8 8 8 }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/20)
    }
  }
}
```



Ma se osserviamo con attenzione possiamo vedere che le note della seconda metà della nonina hanno una spaziatura leggermente più larga delle note della prima parte della nonina. Per assicurare un allungamento uniforme, attiviamo `uniform-stretching`, una proprietà di `SpacingSpanner`.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
    \new RhythmicStaff {
      \tuplet 9/8 { c8 8 8 8 8 8 8 8 8 }
    }
  >>
  \layout {
    \context {
      \Score
      proportionalNotationDuration = #(ly:make-moment 1/20)
      \override SpacingSpanner.uniform-stretching = ##t
    }
  }
}
```



Il nostro esempio di due righe ora ha una spaziatura esatta, le relazioni ritmiche sono visivamente chiare e possiamo includere una linea del tempo con tacche o altro elemento grafico, se lo vogliamo.

Nota che la notazione proporzionale di LilyPond si aspetta che tutte le partiture proporzionali impostino l'attributo `'uniform-stretching` di `SpacingSpanner` su `##t`. Impostare `proportionalNotationDuration` senza impostare anche l'attributo `'uniform-stretching` di `SpacingSpanner` su `##t` farà sì, per esempio, che le pause invisibili occupino una quantità scorretta di spazio orizzontale.

`SpacingSpanner` è un grob astratto che si trova nel contesto `Score`. Come per le impostazioni di `proportionalNotationDuration`, le modifiche `\override` a `SpacingSpanner` si possono trovare in uno dei tre diversi punti del file di input – nel blocco `\with` del brano, nel blocco `\context` del brano o direttamente in mezzo alle note.

Per impostazione predefinita, esiste un solo `SpacingSpanner` per `Score`. Ciò significa che `uniform-stretching` è attivato o disattivato per l'intera partitura. Possiamo tuttavia modificare tale comportamento e attivare diverse funzionalità di spaziatura in punti diversi del brano. Per farlo si usa il comando `\newSpacingSection`. Maggiori informazioni in [\[New spacing area\]](#), pagina [\[undefined\]](#).

Ora esaminiamo gli effetti dell'incisore `Separating_line_group_engraver` e vediamo perché le partiture proporzionali solitamente tolgano questo incisore. L'esempio seguente mostra che c'è un piccolo spazio "introduttivo" proprio prima della prima nota di ogni sistema.

```
\paper {
  indent = #0
}
```

```
\new Staff {
  c'1
  \break
  c'1
}
```



L'ampiezza di questo spazio introduttivo è la stessa sia dopo un'indicazione di tempo che dopo un'armatura di chiave o una chiave. È l'incisore `Separating_line_group_engraver` a causare questo spazio. Togliendolo lo spazio diventa zero.

```
\paper {
  indent = #0
}
```

```
\new Staff \with {
  \remove "Separating_line_group_engraver"
} {
  c'1
  \break
  c'1
}
```



Nella notazione proporzionale gli elementi non musicali come le indicazioni di tempo, le armature di chiave, le chiavi e le alterazioni sono problematiche. Nessuna di queste infatti ha una durata ritmica, ma tutte occupano spazio orizzontale. Questi problemi sono affrontati diversamente dalle varie partiture proporzionali.

È possibile evitare i problemi di spaziatura dovuti alle armature di chiave semplicemente omettendole. Questa è un'opzione valida dato che la maggior parte delle partiture proporzionali sono di musica contemporanea. Lo stesso potrebbe valere per le indicazioni di tempo, specialmente per quelle partiture che includono una linea del tempo o altri elementi grafici. Ma queste partiture sono un'eccezione e la maggior parte delle partiture proporzionali hanno almeno qualche indicazione di tempo. Le chiavi e le alterazioni sono ancora più fondamentali.

Dunque quali strategie adottare per spaziare elementi non musicali nel contesto di musica proporzionale? Una valida opzione è la proprietà `strict-note-spacing` di `SpacingSpanner`. Confrontiamo i seguenti due righi:

```
\new Staff {
  \set Score.proportionalNotationDuration = #(ly:make-moment 1/16)
  c''8 8 8 \clef alto d'2 2
}
```

```
\new Staff {
  \set Score.proportionalNotationDuration = #(ly:make-moment 1/16)
  \override Score.SpacingSpanner.strict-note-spacing = ##t
  c''8 8 8 \clef alto d'2 2
}
```



Entrambi sono proporzionali, ma la spaziatura del primo è troppo larga a causa del cambio di chiave. La spaziatura del secondo resta invece stretta, perché è attivato `strict-note-spacing`. Attivando `strict-note-spacing`, la larghezza di indicazioni di tempo, armature di chiave, chiavi e alterazioni non ha alcun ruolo nell'algoritmo di spaziatura.

Oltre alle impostazioni che abbiamo visto, ce ne sono altre che appaiono frequentemente nelle partiture proporzionali, tra cui:

- `\override SpacingSpanner.strict-grace-spacing = ##t`
- `\set tupletFullLength = ##t`
- `\override Beam.breakable = ##t`
- `\override Glissando.breakable = ##t`
- `\override TextSpanner.breakable = ##t`
- `\remove "Forbid_line_break_engraver"` nel contesto `Voice`

Queste impostazioni spaziano in modo conciso gli abbellimenti, estendono le parentesi dei gruppi irregolari per contrassegnare i punti di inizio e di fine del ritmo, e permettono agli elementi che si estendono orizzontalmente di andare oltre i sistemi e le pagine. Consultare le sezioni del manuale per queste impostazioni.

## Vedi anche

Guida alla notazione: [\[New spacing area\]](#), pagina [\[undefined\]](#).

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

## 4.6 Riduzione del numero di pagine di una partitura

Talvolta può capitare di avere uno o due righi su una seconda (o terza, o quarta. . .) pagina. Ciò è fastidioso, specialmente se c'è molto spazio nelle pagine precedenti.

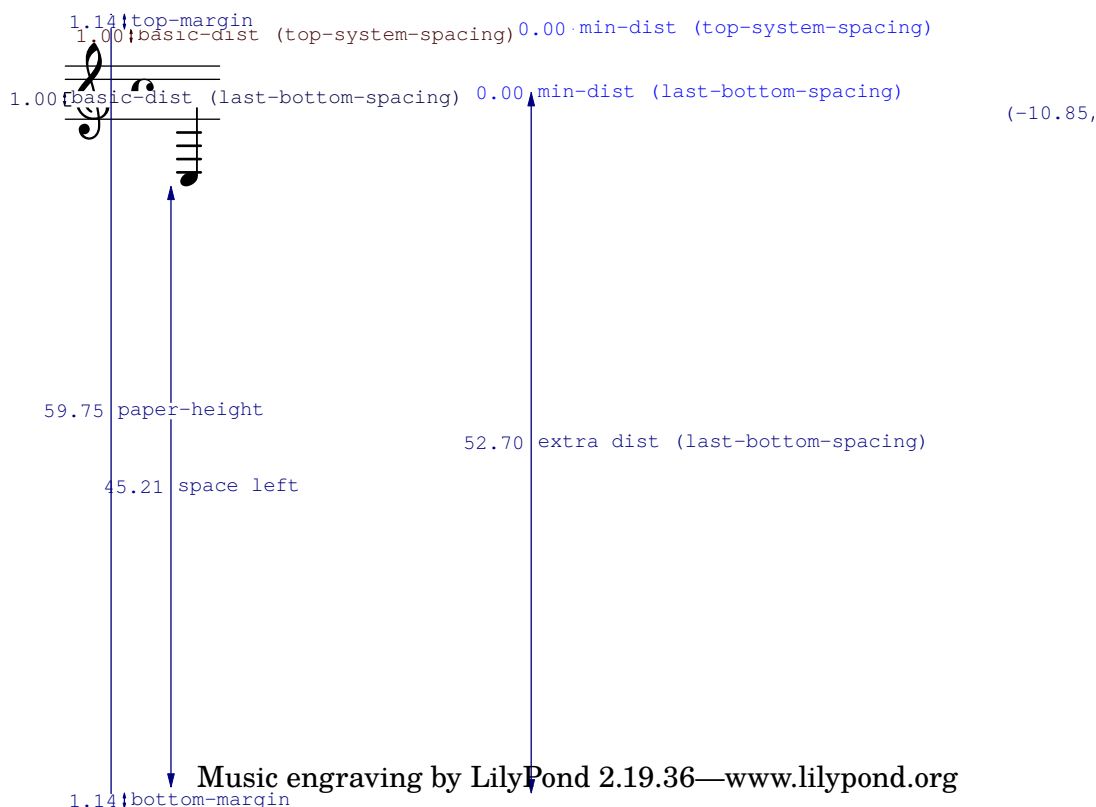
Quando si studiano i problemi di formattazione, uno strumento irrinunciabile è `annotate-spacing`, un comando che mostra i valori delle diverse variabili di spaziatura. Maggiori dettagli nella prossima sezione, [\[Displaying spacing\]](#), pagina [\[undefined\]](#).



### 4.6.1 Visualizzare la spaziatura

Per visualizzare graficamente le dimensioni delle variabili della formattazione verticale che possono essere modificate per formattare la pagina, impostare `annotate-spacing` nel blocco `\paper`:

```
\book {
  \score { { c4 } }
  \paper { annotate-spacing = ##t }
}
```



Tutte le dimensioni della formattazione sono visualizzate in spazi rigo, indipendentemente dalle unità di misura specificate nei blocchi `\paper` o `\layout`. Nell'esempio qui sopra, l'altezza del foglio (`paper-height`) ha un valore di 59.75 spazi rigo (`staff-space`), e la dimensione del rigo (`staff-size`) è pari a 20 punti (il valore predefinito). Nota che:

$$\begin{aligned}
 1 \text{ punto} &= (25.4/72.27) \text{ mm} \\
 1 \text{ staff-space} &= (\text{staff-size})/4 \text{ pts} \\
 &= (\text{staff-size})/4 * \\
 &= (25.4/72.27) \text{ mm}
 \end{aligned}$$

In questo caso, uno `staff-space` è uguale all'incirca a 1.757mm. Dunque i 59.75 `staff-space` di `paper-height` corrispondono a 105 millimetri, pari all'altezza del formato `a6` in orientamento orizzontale. Le coppie  $(a,b)$  sono intervalli, dove  $a$  è l'estremo inferiore e  $b$  l'estremo superiore dell'intervallo.

### Vedi anche

Guida alla notazione: `<undefined>` [Setting the staff size], pagina `<undefined>`.

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

### 4.6.2 Modificare la spaziatura

L'output di `annotate-spacing` svela le dimensioni verticali molto dettagliatamente. Maggiori informazioni su come modificare i margini e altre variabili di formattazione si trovano in `<undefined>` [Page layout], pagina `<undefined>`.

Oltre ai margini, ci sono altre opzioni utili per salvare spazio:

- Forzare i sistemi perché si avvicinino il più possibile (per far entrare più sistemi possibile in una pagina) mentre sono spaziati in modo da non lasciare spazio bianco in fondo alla pagina.

```
\paper {
  system-system-spacing = #'((basic-distance . 0.1) (padding . 0))
  ragged-last-bottom = ##f
  ragged-bottom = ##f
}
```

- Forzare il numero dei sistemi. Ciò può essere utile in due modi. Il semplice impostare un valore, persino lo stesso valore del numero di sistemi disposti senza modificare la variabile, può far sì che più sistemi riescano a entrare in ogni pagina, perché viene saltato il passaggio di valutazione, dando un valore più adatto per ogni pagina. Inoltre, forzare davvero una riduzione nel numero di sistemi può far risparmiare un'ulteriore pagina. Per esempio, se la formattazione predefinita ha 11 sistemi, la seguente impostazione forzerà la formattazione con 10 sistemi.

```
\paper {
  system-count = #10
}
```

- Forzare il numero delle pagine. Per esempio, la seguente impostazione forzerà la formattazione in due pagine.

```
\paper {
  page-count = #2
}
```

- Evitare (o ridurre) gli oggetti che aumentano la dimensione verticale di un sistema. Per esempio, le parentesi delle volte per i finali alternativi richiedono ulteriore spazio. Se questi finali si estendono per due sistemi, occupano più spazio che se fossero sullo stesso sistema. Altro esempio: le dinamiche che “spuntano fuori” da un sistema possono essere avvicinate al rigo:

```
\relative e' {
  e4 c g\ff c
  e4 c g-\tweak X-offset #-2.7 \f c
}
```



- Modificare la spaziatura orizzontale tramite `SpacingSpanner`. Maggiori informazioni in `<undefined>` [Changing horizontal spacing], pagina `<undefined>`. L'esempio seguente mostra la spaziatura predefinita:

```
\score {
  \relative {
    g'4 e e2 |
```

```

f4 d d2 |
c4 d e f |
g4 g g2 |
g4 e e2 |
}
}

```



L'esempio successivo modifica `common-shortest-duration` da un valore di  $1/4$  a uno di  $1/2$ . La nota di un quarto è la durata più comune e più breve in questo esempio, dunque rendendola più lunga si verifica un effetto “compressione”:

```

\score {
  \relative {
    g'4 e e2 |
    f4 d d2 |
    c4 d e f |
    g4 g g2 |
    g4 e e2 |
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner.common-shortest-duration =
        #(ly:make-moment 1/2)
    }
  }
}

```



La proprietà `common-shortest-duration` non può essere modificata in modo dinamico, quindi deve essere sempre posta in un blocco `\context` così che sia applicata all'intera partitura.

## Vedi anche

Guida alla notazione: [\[Page layout\]](#), pagina [\[Changing horizontal spacing\]](#), pagina [\[Changing horizontal spacing\]](#).

Frammenti: Sezione “Spacing” in *Frammenti di codice*.

## 5 Modifica delle impostazioni predefinite

The purpose of LilyPond’s design is to provide the finest quality output by default. Nevertheless, it may happen that you need to change this default layout. The layout is controlled through a large number of ‘knobs and switches’ collectively called ‘properties’. A tutorial introduction to accessing and modifying these properties can be found in the *Manuale di apprendimento*, see Sezione “Modifica dell’output” in *Manuale di Apprendimento*. This should be read first. This chapter covers similar ground, but in a style more appropriate to a reference manual.

The definitive description of the controls available for tuning can be found in a separate document: Sezione “the Internals Reference” in *Guida al Funzionamento Interno*. That manual lists all the variables, functions and options available in LilyPond. It is written as a HTML document, which is available on-line (<http://lilypond.org/doc/stable/Documentation/internals/>), and is also included with the LilyPond documentation package.

Internally, LilyPond uses Scheme (a LISP dialect) to provide infrastructure. Overriding layout decisions in effect accesses the program internals, which requires Scheme input. Scheme elements are introduced in a .ly file with the hash mark #.<sup>1</sup>

### 5.1 Contesti di interpretazione

This section describes what contexts are, and how to modify them.

#### Vedi anche

Manuale di apprendimento: Sezione “Contesti e incisori” in *Manuale di Apprendimento*.

File installati: ly/engraver-init.ly, ly/performer-init.ly.

Frammenti: Sezione “Contexts and engravers” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “Contexts” in *Guida al Funzionamento Interno*, Sezione “Engravers and Performers” in *Guida al Funzionamento Interno*.

#### 5.1.1 Contexts explained

Contexts are arranged hierarchically:

#### Output definitions - blueprints for contexts

This section explains the relevance of output definitions when working with contexts. Examples for actual output definitions are given later (see [Changing all contexts of the same type], pagina 589).

While music written in a file may refer to context types and names, contexts are created only when the music is actually being interpreted. LilyPond interprets music under control of an ‘output definition’ and may do so for several different output definitions, resulting in different output. The output definition relevant for printing music is specified using `\layout`.

A much simpler output definition used for producing Midi output is specified using `\midi`. Several other output definitions are used by LilyPond internally, like when using the part combiner (`\combine`) [Automatic part combining], pagina `\combine`) or creating music quotes (`\quote`) [Quoting other voices], pagina `\quote`).

Output definitions define the relation between contexts as well as their respective default settings. While most changes will usually be made inside of a `\layout` block, Midi-related settings will only have an effect when made within a `\midi` block.

Some settings affect several outputs: for example, if `autoBeaming` is turned off in some context, beams count as melismata for the purpose of matching music to lyrics as described in

<sup>1</sup> Sezione “Scheme tutorial” in *Estendere*, contains a short tutorial on entering numbers, lists, strings, and symbols in Scheme.

`\autoBeaming` [Automatic syllable durations], pagina `\autoBeaming`. This matching is done both for printed output as well as for Midi. If changes made to `\autoBeaming` within a context definition of a `\layout` block are not repeated in the corresponding `\midi` block, lyrics and music will get out of sync in Midi.

## Vedi anche

File installati: `ly/engraver-init.ly`, `ly/performer-init.ly`.

## Score - the master of all contexts

This is the top level notation context. No other context can contain a Score context. By default the Score context handles the administration of time signatures and makes sure that items such as clefs, time signatures, and key-signatures are aligned across staves.

A Score context is instantiated implicitly when a `\score { ... }` block is processed.

## Top-level contexts - staff containers

### *StaffGroup*

Groups staves while adding a bracket on the left side, grouping the staves together. The bar lines of the contained staves are connected vertically. **StaffGroup** only consists of a collection of staves, with a bracket in front and spanning bar lines.

### *ChoirStaff*

Identical to **StaffGroup** except that the bar lines of the contained staves are not connected vertically.

### *GrandStaff*

A group of staves, with a brace on the left side, grouping the staves together. The bar lines of the contained staves are connected vertically.

### *PianoStaff*

Just like **GrandStaff**, but with support for instrument names to the left of each system.

## Intermediate-level contexts - staves

### *Staff*

Handles clefs, bar lines, keys, accidentals. It can contain **Voice** contexts.

### *RhythmicStaff*

Like **Staff** but for printing rhythms. Pitches are ignored when engraving; the notes are printed on one line. The MIDI rendition retains pitches unchanged.

### *TabStaff*

Context for generating tablature. By default lays the music expression out as a guitar tablature, printed on six lines.

### *DrumStaff*

Handles typesetting for percussion. Can contain **DrumVoice**

### *VaticanaStaff*

Same as **Staff**, except that it is designed for typesetting a piece in gregorian style.

### *MensuralStaff*

Same as **Staff**, except that it is designed for typesetting a piece in mensural style.

## Bottom-level contexts - voices

Voice-level contexts initialise certain properties and start appropriate engravers. A bottom-level context is one without `defaultchild`. While it is possible to let it accept/contain subcontexts, they can only be created and entered explicitly.

### *Voice*

Corresponds to a voice on a staff. This context handles the conversion of dynamic signs, stems, beams, super- and sub-scripts, slurs, ties, and rests. You have to instantiate this explicitly if you require multiple voices on the same staff.

### *VaticanaVoice*

Same as **Voice**, except that it is designed for typesetting a piece in gregorian style.

### *MensuralVoice*

Same as **Voice**, with modifications for typesetting a piece in mensural style.

### *Lyrics*

Corresponds to a voice with lyrics. Handles the printing of a single line of lyrics.

### *DrumVoice*

The voice context used in a percussion staff.

### *FiguredBass*

The context in which **BassFigure** objects are created from input entered in `\figuremode` mode.

### *TabVoice*

The voice context used within a **TabStaff** context. Usually left to be created implicitly.

### *CueVoice*

A voice context used to render notes of a reduced size, intended primarily for adding cue notes to a staff, see `<undefined>` [Formatting cue notes], pagina `<undefined>`. Usually left to be created implicitly.

### *ChordNames*

Typesets chord names.

## 5.1.2 Creating and referencing contexts

LilyPond will create lower-level contexts automatically if a music expression is encountered before a suitable context exists, but this is usually successful only for simple scores or music fragments like the ones in the documentation. For more complex scores it is advisable to specify all contexts explicitly with either the `\new` or `\context` command. The syntax of these two commands is very similar:

```
[\new | \context] Context [ = name] [music-expression]
```

where either `\new` or `\context` may be specified. *Context* is the type of context which is to be created, *name* is an optional name to be given to the particular context being created and *music-expression* is a single music expression that is to be interpreted by the engravers and performers in this context.

The `\new` prefix without a name is commonly used to create scores with many staves:

```
<<
\new Staff \relative {
  % leave the Voice context to be created implicitly
  c''4 c
}
\new Staff \relative {
```

```

    d''4 d
  }
>>

```



and to place several voices into one staff:

```

\new Staff <<
  \new Voice \relative {
    \voiceOne
    c''8 c c4 c c
  }
  \new Voice \relative {
    \voiceTwo
    g'4 g g g
  }
>>

```



`\new` should always be used to specify unnamed contexts.

The difference between `\new` and `\context` is in the action taken:

- `\new` with or without a name will always create a fresh, distinct, context, even if one with the same name already exists:

```

\new Staff <<
  \new Voice = "A" \relative {
    \voiceOne
    c''8 c c4 c c
  }
  \new Voice = "A" \relative {
    \voiceTwo
    g'4 g g g
  }
>>

```



- `\context` with a name specified will create a distinct context only if a context of the same type with the same name in the same context hierarchy does not already exist. Otherwise it will be taken as a reference to that previously created context, and its music expression will be passed to that context for interpretation.

One application of named contexts is in separating the score layout from the musical content. Either of these two forms is valid:

```
\score {
  <<
    % score layout
    \new Staff <<
      \new Voice = "one" {
        \voiceOne
      }
      \new Voice = "two" {
        \voiceTwo
      }
    >>

    % musical content
    \context Voice = "one" {
      \relative {
        c''4 c c c
      }
    }
    \context Voice = "two" {
      \relative {
        g'8 g g4 g g
      }
    }
  >>
}
```



```
\score {
  <<
    % score layout
    \new Staff <<
      \context Voice = "one" {
        \voiceOne
      }
      \context Voice = "two" {
        \voiceTwo
      }
    >>

    % musical content
    \context Voice = "one" {
      \relative {
        c''4 c c c
      }
    }
    \context Voice = "two" {
      \relative {

```



```

      g'8 g g4 g g
    }
  }
  >>
}

```



Alternatively, variables may be employed to similar effect. See Sezione “Organizzare i brani con le variabili” in *Manuale di Apprendimento*.

- `\context` with no name will match the first of any previously created contexts of the same type in the same context heirarchy, even one that has been given a name, and its music expression will be passed to that context for interpretation. This form is rarely useful. However, `\context` with no name and no music expression is used to set the context in which a Scheme procedure specified with `\applyContext` is executed:

```

\new Staff \relative {
  c'1
  \context Timing
  \applyContext #(lambda (ctx)
                    (newline)
                    (display (ly:context-current-moment ctx)))
  c1
}

```

A context must be named if it is to be referenced later, for example when lyrics are associated with music:

```

\new Voice = "tenor" music
...
\new Lyrics \lyricsto "tenor" lyrics

```

For details of associating lyrics with music see [\[Automatic syllable durations\]](#), pagina [\[undefined\]](#).

The properties of all contexts of a particular type can be modified in a `\layout` block (with a different syntax), see [\[Changing all contexts of the same type\]](#), pagina 589. This construct also provides a means of keeping layout instructions separate from the musical content. If a single context is to be modified, a `\with` block must be used, see [\[Changing just one specific context\]](#), pagina 592.

## Vedi anche

Manuale di apprendimento: Sezione “Organizzare i brani con le variabili” in *Manuale di Apprendimento*.

Guida alla notazione: [\[Changing just one specific context\]](#), pagina 592, [\[Automatic syllable durations\]](#), pagina [\[undefined\]](#).

### 5.1.3 Keeping contexts alive

Contexts are usually terminated at the first musical moment in which they have nothing to do. So **Voice** contexts die as soon as they contain no events; **Staff** contexts die as soon as all the **Voice** contexts within them contain no events; etc. This can cause difficulties if earlier contexts which have died have to be referenced, for example, when changing staves with `\change`

commands, associating lyrics with a voice with `\lyricsto` commands, or when adding further musical events to an earlier context.

There is an exception to this general rule: inside of an `{...}` construct (sequential music), the construct's notion of the "current context" will descend whenever an element of the sequence ends in a subcontext of the previous current context. This avoids spurious creation of implicit contexts in a number of situations but means that the first context descended into will be kept alive until the end of the expression.

In contrast, the contexts of a `<<...>>` construct's (simultaneous music) expression are not carried forth, so enclosing a context creating command in an extra pair of `<<...>>` will keep the context from persisting through all of the enclosing `{...}` sequence.

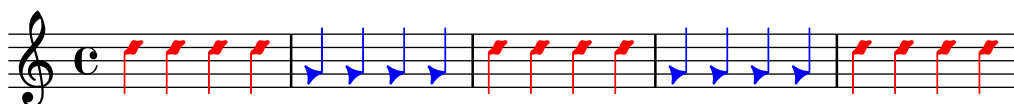
Any context can be kept alive by ensuring it has something to do at every musical moment. **Staff** contexts are kept alive by ensuring one of their voices is kept alive. One way of doing this is to add spacer rests to a voice in parallel with the real music. These need to be added to every **Voice** context which needs to be kept alive. If several voices are to be used sporadically it is safest to keep them all alive rather than attempting to rely on the exceptions mentioned above.

In the following example, both voice A and voice B are kept alive in this way for the duration of the piece:

```
musicA = \relative { d''4 d d d }
musicB = \relative { g'4 g g g }
keepVoicesAlive = {
  <<
    \new Voice = "A" { s1*5 } % Keep Voice "A" alive for 5 bars
    \new Voice = "B" { s1*5 } % Keep Voice "B" alive for 5 bars
  >>
}

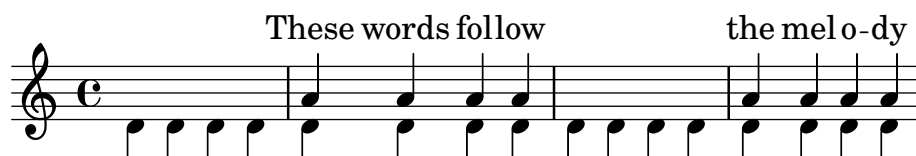
music = {
  \context Voice = "A" {
    \voiceOneStyle
    \musicA
  }
  \context Voice = "B" {
    \voiceTwoStyle
    \musicB
  }
  \context Voice = "A" { \musicA }
  \context Voice = "B" { \musicB }
  \context Voice = "A" { \musicA }
}

\score {
  \new Staff <<
    \keepVoicesAlive
    \music
  >>
}
```



The following example shows how a sporadic melody line with lyrics might be written using this approach. In a real situation the melody and accompaniment would consist of several different sections, of course.

```
melody = \relative { a'4 a a a }
accompaniment = \relative { d'4 d d d }
words = \lyricmode { These words fol -- low the mel -- o -- dy }
\score {
  <<
    \new Staff = "music" {
      <<
        \new Voice = "melody" {
          \voiceOne
          s1*4 % Keep Voice "melody" alive for 4 bars
        }
        {
          \new Voice = "accompaniment" {
            \voiceTwo
            \accompaniment
          }
          <<
            \context Voice = "melody" { \melody }
            \context Voice = "accompaniment" { \accompaniment }
          >>
          \context Voice = "accompaniment" { \accompaniment }
          <<
            \context Voice = "melody" { \melody }
            \context Voice = "accompaniment" { \accompaniment }
          >>
        }
      >>
    }
    \new Lyrics \with { alignAboveContext = #"music" }
    \lyricsto "melody" { \words }
  >>
}
```



An alternative way, which may be better in many circumstances, is to keep the melody line alive by simply including spacer notes to line it up correctly with the accompaniment:

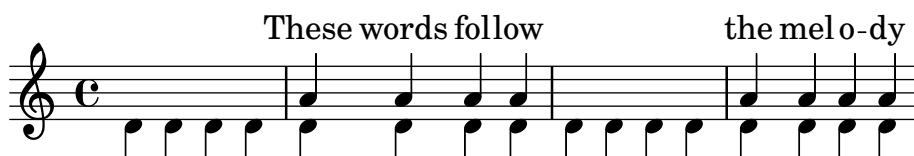
```
melody = \relative {
  s1 % skip a bar
  a'4 a a a
  s1 % skip a bar
  a4 a a a
}
accompaniment = \relative {
  d'4 d d d
```

```

d4 d d d
d4 d d d
d4 d d d
}
words = \lyricmode { These words fol -- low the mel -- o -- dy }

\score {
  <<
    \new Staff = "music" {
      <<
        \new Voice = "melody" {
          \voiceOne
          \melody
        }
        \new Voice = "accompaniment" {
          \voiceTwo
          \accompaniment
        }
      >>
    }
    \new Lyrics \with { alignAboveContext = #"music" }
    \lyricsto "melody" { \words }
  >>
}

```



#### 5.1.4 Modifying context plug-ins

Notation contexts (like `Score` and `Staff`) not only store properties, they also contain plug-ins called ‘engravers’ that create notation elements. For example, the `Voice` context contains a `Note_heads_engraver` and the `Staff` context contains a `Key_engraver`.

For a full a description of each plug-in, see Guida al funzionamento interno  $\mapsto$  Translation  $\mapsto$  Engravers. Every context described in Guida al funzionamento interno  $\mapsto$  Translation  $\mapsto$  Context. lists the engravers used for that context.

It can be useful to shuffle around these plug-ins. This is done by starting a new context with `\new` or `\context`, and modifying it,

```

\new context \with {
  \consists ...
  \consists ...
  \remove ...
  \remove ...
  etc.
}
{
  ...music...
}

```

where the `...` should be the name of an engraver. Here is a simple example which removes `Time_signature_engraver` and `Clef_engraver` from a `Staff` context,

```
<<
\new Staff \relative {
  f'2 g
}
\new Staff \with {
  \remove "Time_signature_engraver"
  \remove "Clef_engraver"
} \relative {
  f'2 g2
}
>>
```



In the second staff there are no time signature or clef symbols. This is a rather crude method of making objects disappear since it will affect the entire staff. This method also influences the spacing, which may or may not be desirable. More sophisticated methods of blanking objects are shown in Sezione “Visibilità e colore degli oggetti” in *Manuale di Apprendimento*.

The next example shows a practical application. Bar lines and time signatures are normally synchronized across the score. This is done by the `Timing_translator` and `Default_bar_line_engraver`. This plug-in keeps an administration of time signature, location within the measure, etc. By moving these engraver from `Score` to `Staff` context, we can have a score where each staff has its own time signature.

```
\score {
  <<
    \new Staff \with {
      \consists "Timing_translator"
      \consists "Default_bar_line_engraver"
    }
    \relative {
      \time 3/4
      c''4 c c c c c
    }
  \new Staff \with {
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  }
  \relative {
    \time 2/4
    c''4 c c c c c
  }
  >>
\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Default_bar_line_engraver"
  }
}
```

```

    }
  }
}

```



## Problemi noti e avvertimenti

The order in which the engravers are specified is the order in which they are called to carry out their processing. Usually the order in which the engravers are specified does not matter, but in a few special cases the order is important, for example where one engraver writes a property and another reads it, or where one engraver creates a grob and another must process it.

The following orderings are important:

- the `Bar_engraver` must normally be first,
- the `New_fingering_engraver` must come before the `Script_column_engraver`,
- the `Timing_translator` must come before the `Bar_number_engraver`.

## Vedi anche

File installati: `ly/engraver-init.ly`.

### 5.1.5 Changing context default settings

Context and grob properties can be changed with `\set` and `\override` commands, as described in [\[Modifying properties\]](#), pagina [\[Modifying properties\]](#). These commands create music events, making the changes take effect at the point in time the music is being processed.

In contrast, this section explains how to change the *default* values of context and grob properties at the time the context is created. There are two ways of doing this. One modifies the default values in all contexts of a particular type, the other modifies the default values in just one particular instance of a context.

### Changing all contexts of the same type

The default context settings which are to be used for typesetting in `Score`, `Staff`, `Voice` and other contexts may be specified in a `\context` block within any `\layout` block.

Settings for Midi output as opposed to typesetting will have to be separately specified in `\midi` blocks (see [\[Output definitions - blueprints for contexts\]](#), pagina 579).

The `\layout` block should be placed within the `\score` block to which it is to apply, after the music.

```

\layout {
  \context {
    \Voice
    [context settings for all Voice contexts]
  }
  \context {
    \Staff
    [context settings for all Staff contexts]
  }
}

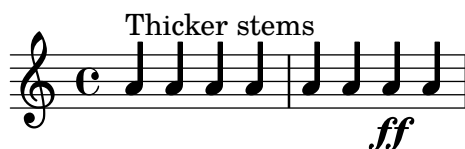
```

```
}
}
```

The following types of settings may be specified:

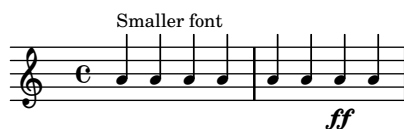
- An `\override` command, but with the context name omitted

```
\score {
  \relative {
    a'4~"Thicker stems" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Staff
      \override Stem.thickness = #4.0
    }
  }
}
```



- Directly setting a context property

```
\score {
  \relative {
    a'4~"Smaller font" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Staff
      fontSize = #-4
    }
  }
}
```



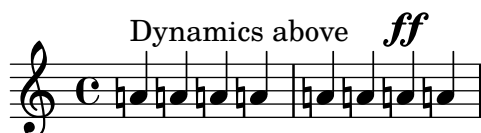
- A predefined command such as `\dynamicUp` or a music expression like `\accidentalStyle dodecaphonic`

```
\score {
  \relative {
    a'4~"Dynamics above" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Voice
      \dynamicUp
    }
  }
}
```

```

    }
    \context {
      \Staff
      \accidentalStyle dodecaphonic
    }
  }
}

```



- A user-defined variable containing a `\with` block; for details of the `\with` block see [Changing just one specific context], pagina 592.

```

StaffDefaults = \with {
  fontSize = #-4
}

\score {
  \new Staff {
    \relative {
      a'4^"Smaller font" a a a
      a4 a a a
    }
  }
  \layout {
    \context {
      \Staff
      \StaffDefaults
    }
  }
}

```



Property-setting commands can be placed in a `\layout` block without being enclosed in a `\context` block. Such settings are equivalent to including the same property-setting commands at the start of every context of the type specified. If no context is specified *every* bottom-level context is affected, see [Bottom-level contexts - voices], pagina 581. The syntax of a property-setting command in a `\layout` block is the same as the same command written in the music stream.

```

\score {
  \new Staff {
    \relative {
      a'4^"Smaller font" a a a
      a4 a a a
    }
  }
  \layout {

```



```

\accidentalStyle dodecaponic
\set fontSize = #-4
\override Voice.Stem.thickness = #4.0
}
}

```



## Changing just one specific context

The context properties of just one specific context instance can be changed in a `\with` block. All other context instances of the same type retain the default settings built into LilyPond and modified by any `\layout` block within scope. The `\with` block must be placed immediately after the `\new context-type` command:

```

\new Staff \with { [context settings for this context instance only] }
{
  ...
}

```

Since such a ‘context modification’ is specified inside of music, it will affect *all* outputs (typesetting *and* Midi) as opposed to changes within an output definition.

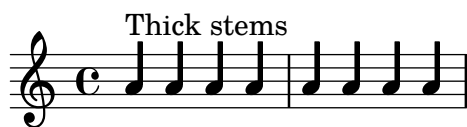
The following types of settings may be specified:

- An `\override` command, but with the context name omitted

```

\score {
  \new Staff {
    \new Voice \with { \override Stem.thickness = #4.0 }
    {
      \relative {
        a'4~"Thick stems" a a a
        a4 a a a
      }
    }
  }
}

```



- Directly setting a context property

```

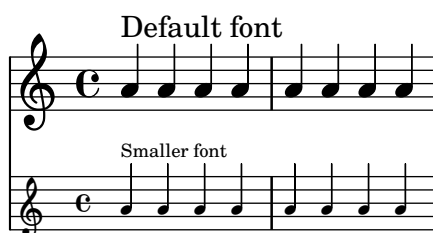
\score {
  <<
    \new Staff {
      \relative {
        a'4~"Default font" a a a
        a4 a a a
      }
    }
    \new Staff \with { fontSize = #-4 }
  {

```

```

\relative {
  a'4~"Smaller font" a a a
  a4 a a a
}
}
>>
}

```

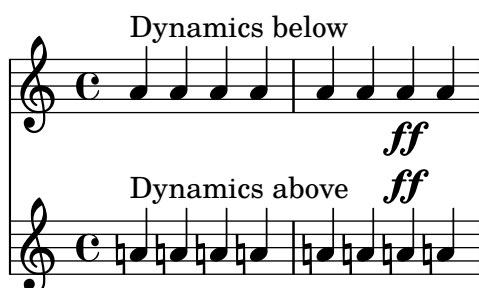


- A predefined command such as `\dynamicUp`

```

\score {
  <<
    \new Staff {
      \new Voice {
        \relative {
          a'4~"Dynamics below" a a a
          a4 a a\ff a
        }
      }
    }
    \new Staff \with { \accidentalStyle dodecaphonic }
    {
      \new Voice \with { \dynamicUp }
      {
        \relative {
          a'4~"Dynamics above" a a a
          a4 a a\ff a
        }
      }
    }
  >>
}

```



## Order of precedence

The value of a property which applies at a particular time is determined as follows:

- if an `\override` or `\set` command in the input stream is in effect that value is used,

- otherwise the default value taken from a `\with` statement on the context initiation statement is used,
- otherwise the default value taken from the most recent appropriate `\context` block in the `\layout` or `\midi` blocks is used,
- otherwise the LilyPond built-in default is used.

## Vedi anche

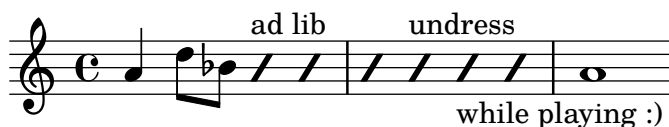
Manuale di apprendimento: Sezione “Modificare le proprietà di contesto” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.1.1 [Contexts explained], pagina 579, [Bottom-level contexts - voices], pagina 581, Sezione 5.3.2 [The set command], pagina 601, Sezione 5.3.3 [The override command], pagina 603, `\context` [The `\layout` block], pagina `\context`.

### 5.1.6 Defining new contexts

Specific contexts, like **Staff** and **Voice**, are made from simple building blocks. It is possible to create new types of contexts with different combinations of engraver plug-ins.

The next example shows how to build a different type of **Voice** context from scratch. It will be similar to **Voice**, but only prints centered slash note heads. It can be used to indicate improvisation in jazz pieces,



These settings are defined within a `\context` block inside a `\layout` block,

```
\layout {
  \context {
    ...
  }
}
```

In the following discussion, the example input shown should go in place of the `...` in the previous fragment.

First it is necessary to define a name for the new context:

```
\name ImproVoice
```

Since it is similar to the **Voice** context, we want commands that work in (existing) **Voice** contexts to continue working. This is achieved by giving the new context an alias of **Voice**,

```
\alias Voice
```

The context will print notes and instructive texts, so we need to add the engravers which provide this functionality, plus the engraver which groups notes, stems and rests which occur at the same musical moment into columns,

```
\consists "Note_heads_engraver"
\consists "Text_engraver"
\consists "Rhythmic_column_engraver"
```

The note heads should all be placed on the center line,

```
\consists "Pitch_squash_engraver"
squashedPosition = #0
```

The `Pitch_squash_engraver` modifies note heads (created by the `Note_heads_engraver`) and sets their vertical position to the value of `squashedPosition`, in this case 0, the center line.

The notes look like a slash, and have no stem,

```
\override NoteHead.style = #'slash
\hide Stem
```

All these plug-ins have to communicate under the control of the context. The mechanisms with which contexts communicate are established by declaring the context `\type`. Within a `\layout` block, most contexts will be of type `Engraver_group`. Some special contexts and contexts in `\midi` blocks use other context types. Copying and modifying an existing context definition will also fill in the type. Since this example creates a definition from scratch, it needs to be specified explicitly.

```
\type "Engraver_group"
```

Put together, we get

```
\context {
  \name ImproVoice
  \type "Engraver_group"
  \consists "Note_heads_engraver"
  \consists "Text_engraver"
  \consists "Rhythmic_column_engraver"
  \consists "Pitch_squash_engraver"
  squashedPosition = #0
  \override NoteHead.style = #'slash
  \hide Stem
  \alias Voice
}
```

Contexts form hierarchies. We want to place the `ImproVoice` context within the `Staff` context, just like normal `Voice` contexts. Therefore, we modify the `Staff` definition with the `\accepts` command,

```
\context {
  \Staff
  \accepts ImproVoice
}
```

Often when reusing an existing context definition, the resulting context can be used anywhere where the original context would have been useful.

```
\layout {
  ...
  \inherit-acceptability to from
}
```

will arrange to have contexts of type *to* accepted by all contexts also accepting *from*. For example, using

```
\layout {
  ...
  \inherit-acceptability "ImproVoice" "Voice"
}
```

will add an `\accepts` for `ImproVoice` to both `Staff` and `RhythmicStaff` definitions.

The opposite of `\accepts` is `\denies`, which is sometimes needed when reusing existing context definitions.

Arranging the required pieces into a `\layout` block leaves us with

```
\layout {
  \context {
```

```

    \name ImproVoice
    ...
}
\inherit-acceptability "ImproVoice" "Voice"
}

```

Then the output at the start of this subsection can be entered as

```

\relative {
  a'4 d8 bes8
  \new ImproVoice {
    c4^"ad lib" c
    c4 c^"undress"
    c c_"while playing :)"
  }
  a1
}

```

To complete this example, changes affecting the context hierarchy should be repeated in a `\midi` block so that Midi output depends on the same context relations.

## Vedi anche

Guida al funzionamento interno: Sezione “`Note_heads_engraver`” in *Guida al Funzionamento Interno*, Sezione “`Text_engraver`” in *Guida al Funzionamento Interno*, Sezione “`Rhythmic_column_engraver`” in *Guida al Funzionamento Interno*, Sezione “`Pitch_squash_engraver`” in *Guida al Funzionamento Interno*.

### 5.1.7 Context layout order

Contexts are normally positioned in a system from top to bottom in the order in which they are encountered in the input file. When contexts are nested, the outer context will include inner nested contexts as specified in the input file, provided the inner contexts are included in the outer context’s “accepts” list. Nested contexts which are not included in the outer context’s “accepts” list will be repositioned below the outer context rather than nested within it.

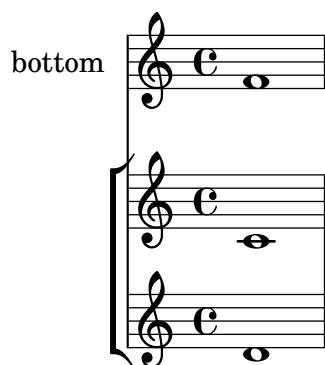
The “accepts” list of a context can be changed with the `\accepts` or `\denies` commands. `\accepts` adds a context to the “accepts” list and `\denies` removes a context from the list.

For example, a square-braced staff group is not usually found within a curved-braced staff with connecting staff bars, and a `GrandStaff` does not accept a `StaffGroup` inside it by default.

```

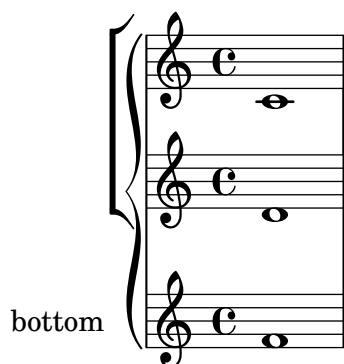
\score {
  \new GrandStaff <<
    \new StaffGroup <<
      \new Staff { c'1 }
      \new Staff { d'1 }
    >>
    \new Staff { \set Staff.instrumentName = bottom f'1 }
  >>
}

```



However, by using the `\accepts` command, `StaffGroup` can be added to the `GrandStaff` context:

```
\score {
  \new GrandStaff <<
    \new StaffGroup <<
      \new Staff { c'1 }
      \new Staff { d'1 }
    >>
    \new Staff { \set Staff.instrumentName = bottom f'1 }
  >>
  \layout {
    \context {
      \GrandStaff
      \accepts "StaffGroup"
    }
  }
}
```



`\denies` is mainly used when a new context is being based on another, but the required nesting differs. For example, the `VaticanaStaff` context is based on the `Staff` context, but with the `VaticanaVoice` context substituted for the `Voice` context in the “accepts” list.

Note that a context will be silently created implicitly if a command is encountered when there is no suitable context available to contain it.

Within a context definition, the type of subcontext to be implicitly created is specified using `\defaultchild`. A number of music events require a ‘`Bottom`’ context: when such an event is encountered, subcontexts are created recursively until reaching a context with no ‘`defaultchild`’ setting.

Implicit context creation can at times give rise to unexpected new staves or scores. Using `\new` to create contexts explicitly avoids those problems.

Sometimes a context is required to exist for just a brief period, a good example being the staff context for an *ossia*. This is usually achieved by introducing the context definition at the

appropriate place in parallel with corresponding section of the main music. By default, the temporary context will be placed below all the existing contexts. To reposition it above the context called “main”, it should be defined like this:

```
\new Staff \with { alignAboveContext = #"main" }
```

A similar situation arises when positioning a temporary lyrics context within a multi-staved layout such as a **ChoirStaff**, for example, when adding a second verse to a repeated section. By default the temporary lyrics context will be placed beneath the lower staves. By defining the temporary lyrics context with **alignBelowContext** it can be positioned correctly beneath the (named) lyrics context containing the first verse.

Examples showing this repositioning of temporary contexts can be found elsewhere — see Sezione “Annidare le espressioni musicali” in *Manuale di Apprendimento*, [\[Modifying single staves\]](#), pagina [\[Techniques specific to lyrics\]](#), pagina [\[Techniques specific to lyrics\]](#).

## Vedi anche

Manuale di apprendimento: Sezione “Annidare le espressioni musicali” in *Manuale di Apprendimento*.

Guida alla notazione: [\[Modifying single staves\]](#), pagina [\[Techniques specific to lyrics\]](#), pagina [\[Techniques specific to lyrics\]](#).

Application Usage: Sezione “An extra staff appears” in *Uso del Programma*.

File installati: `ly/engraver-init.ly`.

## 5.2 Come funziona la Guida al funzionamento interno

### 5.2.1 Navigating the program reference

Suppose we want to move the fingering indication in the fragment below:

c''-2



If you visit the documentation on fingering instructions (in [\[Fingering instructions\]](#), pagina [\[Fingering instructions\]](#)), you will notice:

#### See also

Guida al funzionamento interno: Sezione “Fingering” in *Guida al Funzionamento Interno*.

The programmer’s reference is available as an HTML document. It is highly recommended that you read it in HTML form, either online or by downloading the HTML documentation. This section will be much more difficult to understand if you are using the PDF manual.

Follow the link to Sezione “Fingering” in *Guida al Funzionamento Interno*. At the top of the page, you will see

Fingering objects are created by: Sezione “Fingering\_engraver” in *Guida al Funzionamento Interno* and Sezione “New\_fingering\_engraver” in *Guida al Funzionamento Interno*.

By following related links inside the program reference, we can follow the flow of information within the program:

- Sezione “Fingering” in *Guida al Funzionamento Interno*: Sezione “Fingering” in *Guida al Funzionamento Interno* objects are created by: Sezione “Fingering\_engraver” in *Guida al Funzionamento Interno*

- Sezione “Fingering\_engraver” in *Guida al Funzionamento Interno*: Music types accepted: Sezione “fingering-event” in *Guida al Funzionamento Interno*
- Sezione “fingering-event” in *Guida al Funzionamento Interno*: Music event type `fingering-event` is in Music expressions named Sezione “FingeringEvent” in *Guida al Funzionamento Interno*

This path goes against the flow of information in the program: it starts from the output, and ends at the input event. You could also start at an input event, and read with the flow of information, eventually ending up at the output object(s).

The program reference can also be browsed like a normal document. It contains chapters on Music definitions on Sezione “Translation” in *Guida al Funzionamento Interno*, and the Sezione “Backend” in *Guida al Funzionamento Interno*. Every chapter lists all the definitions used and all properties that may be tuned.

### 5.2.2 Layout interfaces

The HTML page that we found in the previous section describes the layout object called Sezione “Fingering” in *Guida al Funzionamento Interno*. Such an object is a symbol within the score. It has properties that store numbers (like thicknesses and directions), but also pointers to related objects. A layout object is also called a *Grob*, which is short for Graphical Object. For more details about Grobs, see Sezione “grob-interface” in *Guida al Funzionamento Interno*.

The page for **Fingering** lists the definitions for the **Fingering** object. For example, the page says

`padding` (dimension, in staff space):

0.5

which means that the number will be kept at a distance of at least 0.5 of the note head.

Each layout object may have several functions as a notational or typographical element. For example, the Fingering object has the following aspects

- Its size is independent of the horizontal spacing, unlike slurs or beams.
- It is a piece of text. Granted, it is usually a very short text.
- That piece of text is typeset with a font, unlike slurs or beams.
- Horizontally, the center of the symbol should be aligned to the center of the note head.
- Vertically, the symbol is placed next to the note and the staff.
- The vertical position is also coordinated with other superscript and subscript symbols.

Each of these aspects is captured in so-called *interfaces*, which are listed on the Sezione “Fingering” in *Guida al Funzionamento Interno* page at the bottom

This object supports the following interfaces: Sezione “item-interface” in *Guida al Funzionamento Interno*, Sezione “self-alignment-interface” in *Guida al Funzionamento Interno*, Sezione “side-position-interface” in *Guida al Funzionamento Interno*, Sezione “text-interface” in *Guida al Funzionamento Interno*, Sezione “text-script-interface” in *Guida al Funzionamento Interno*, Sezione “font-interface” in *Guida al Funzionamento Interno*, Sezione “finger-interface” in *Guida al Funzionamento Interno*, and Sezione “grob-interface” in *Guida al Funzionamento Interno*.

Clicking any of the links will take you to the page of the respective object interface. Each interface has a number of properties. Some of them are not user-serviceable (‘Internal properties’), but others can be modified.

We have been talking of *the Fingering* object, but actually it does not amount to much. The initialization file (see Sezione “Altre fonti di informazione” in *Manuale di Apprendimento*) `scm/define-grobs.scm` shows the soul of the ‘object’,

(**Fingering**



```
. ((padding . 0.5)
  (avoid-slur . around)
  (slur-padding . 0.2)
  (staff-padding . 0.5)
  (self-alignment-X . 0)
  (self-alignment-Y . 0)
  (script-priority . 100)
  (stencil . ,ly:text-interface::print)
  (direction . ,ly:script-interface::calc-direction)
  (font-encoding . fetaText)
  (font-size . -5) ; don't overlap when next to heads.
  (meta . ((class . Item)
    (interfaces . (finger-interface
      font-interface
      text-script-interface
      text-interface
      side-position-interface
      self-alignment-interface
      item-interface))))))
```

As you can see, the **Fingering** object is nothing more than a bunch of variable settings, and the webpage in the Guida al funzionamento interno is directly generated from this definition.

### 5.2.3 Determining the grob property

Recall that we wanted to change the position of the **2** in

`c''-2`



Since the **2** is vertically positioned next to its note, we have to meddle with the interface associated with this positioning. This is done using **side-position-interface**. The page for this interface says

**side-position-interface**

Position a victim object (this one) next to other objects (the support). The property **direction** signifies where to put the victim object relative to the support (left or right, up or down?)

Below this description, the variable **padding** is described as

**padding** (dimension, in staff space)

Add this much extra space between objects that are next to each other.

By increasing the value of **padding**, we can move the fingering away from the note head. The following command inserts 3 staff spaces of white between the note and the fingering:

```
\once \override Voice.Fingering.padding = #3
```

Inserting this command before the **Fingering** object is created, i.e., before `c2`, yields the following result:

```
\once \override Voice.Fingering.padding = #3
```

`c''-2`



In this case, the context for this tweak is **Voice**. This fact can also be deduced from the program reference, for the page for the Sezione “Fingering\_engraver” in *Guida al Funzionamento Interno* plug-in says

Fingering\_engraver is part of contexts: . . . Sezione “Voice” in *Guida al Funzionamento Interno*

### 5.2.4 Naming conventions

Another thing that is needed, is an overview of the various naming conventions:

- scheme functions: lowercase-with-hyphens (incl. one-word names)
- scheme functions: ly:plus-scheme-style
- music events, music classes and music properties: as-scheme-functions
- Grob interfaces: scheme-style
- backend properties: scheme-style (but X and Y!)
- contexts (and MusicExpressions and grobs): Capitalized or CamelCase
- context properties: lowercaseFollowedByCamelCase
- engravers: Capitalized\_followed\_by\_lowercase\_and\_with\_underscores

Questions to be answered:

- Which of these are conventions and which are rules?
- Which are rules of the underlying language, and which are LP-specific?

## 5.3 Modifica delle proprietà

### 5.3.1 Overview of modifying properties

Each context is responsible for creating certain types of graphical objects. The settings used for printing these objects are also stored by context. By changing these settings, the appearance of objects can be altered.

There are two different kinds of properties stored in contexts: context properties and grob properties. Context properties are properties that apply to the context as a whole and control how the context itself is displayed. In contrast, grob properties apply to specific grob types that will be displayed in the context.

The `\set` and `\unset` commands are used to change values for context properties. The `\override` and `\revert` commands are used to change values for grob properties.

### Vedi anche

Guida al funzionamento interno: Sezione “Backend” in *Guida al Funzionamento Interno*, Sezione “All layout objects” in *Guida al Funzionamento Interno*, Sezione “OverrideProperty” in *Guida al Funzionamento Interno*, Sezione “RevertProperty” in *Guida al Funzionamento Interno*, Sezione “PropertySet” in *Guida al Funzionamento Interno*.

### Problemi noti e avvertimenti

The back-end is not very strict in type-checking object properties. Cyclic references in Scheme values for properties can cause hangs or crashes, or both.

### 5.3.2 The `\set` command

Each context has a set of *properties*, variables contained in that context. Context properties are changed with the `\set` command, which has the following syntax:

```
\set context.property = #value
```

*value* is a Scheme object, which is why it must be preceded by the `#` character.

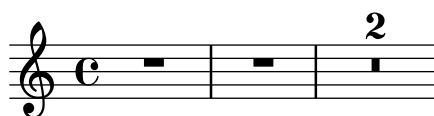
Contexts properties are usually named in **studlyCaps**. They mostly control the translation from music to notation, e.g. **localAlterations** (for determining whether to print accidentals), or **measurePosition** (for determining when to print a bar line). Context properties can change value over time while interpreting a piece of music; **measurePosition** is an obvious example of this. Context properties are modified with `\set`.

For example, multimeasure rests will be combined into a single bar if the context property **skipBars** is set to `##t`:

R1\*2

```
\set Score.skipBars = ##t
```

R1\*2



If the *context* argument is left out, then the property will be set in the current bottom context (typically **ChordNames**, **Voice**, **TabVoice**, or **Lyrics**).

```
\set Score.autoBeaming = ##f
```

```
\relative {
```

```
  e''8 e e e
```

```
  \set autoBeaming = ##t
```

```
  e8 e e e
```

```
} \\\
```

```
\relative {
```

```
  c''8 c c c c8 c c c
```

```
}
```



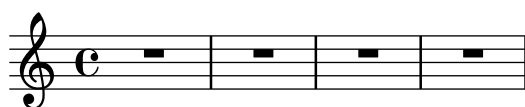
The change is applied ‘on-the-fly’, during the music, so that the setting only affects the second group of eighth notes.

Note that the bottom-most context does not always contain the property that you wish to change – for example, attempting to set the **skipBars** property of the default bottom context, in this case **Voice**, will have no effect, because **skipBars** is a property of the **Score** context.

R1\*2

```
\set skipBars = ##t
```

R1\*2



Contexts are hierarchical, so if an enclosing context was specified, for example **Staff**, then the change would also apply to all **Voices** in the current staff.

The `\unset` command:

```
\unset context.property
```

is used to remove the definition of *property* from *context*. This command removes the definition only if it is set in *context*. Properties that have been set in enclosing contexts will not be altered by an `\unset` in an enclosed context:

```
\set Score.autoBeaming = ##t
```

```

\relative {
  \unset autoBeaming
  e''8 e e e
  \unset Score.autoBeaming
  e8 e e e
} \
\relative {
  c''8 c c c c8 c c c
}

```



Like `\set`, the *context* argument does not have to be specified for a bottom context, so the two statements

```

\set Voice.autoBeaming = ##t
\set autoBeaming = ##t

```

are equivalent if the current bottom context is *Voice*.

Preceding a `\set` or `\unset` command by `\once` makes the setting apply to only a single time-step:

```

c''4
\once \set fontSize = #4.7
c''4
c''4

```



A full description of all available context properties is in the Guida al funzionamento interno, see Translation  $\mapsto$  Tunable context properties.

## Vedi anche

Guida al funzionamento interno: Sezione “Tunable context properties” in *Guida al Funzionamento Interno*.

### 5.3.3 The `\override` command

There is a special type of context property: the grob description. Grob descriptions are named in **StudlyCaps** (starting with capital letters). They contain the ‘default settings’ for a particular kind of grob as an association list. See `scm/define-grobs.scm` to see the settings for each grob description. Grob descriptions are modified with `\override`.

The syntax for the `\override` command is

```

\override [context.]GrobName.property = #value

```

For example, we can increase the thickness of a note stem by overriding the **thickness** property of the **Stem** object:

```

c''4 c''
\override Voice.Stem.thickness = #3.0
c''4 c''

```



If no context is specified in an `\override`, the bottom context is used:

```
\override Staff.Stem.thickness = #3.0
<<
  \relative {
    e''4 e
    \override Stem.thickness = #0.5
    e4 e
  } \\\
  \relative {
    c''4 c c c
  }
>>
```



Some tweakable options are called ‘subproperties’ and reside inside properties. To tweak those, use commands in the form

```
\override Stem.details.beamed-lengths = #'(4 4 3)
```

or to modify the ends of spanners, use a form like these

```
\override TextSpanner.bound-details.left.text = #"left text"
\override TextSpanner.bound-details.right.text = #"right text"
```

The effects of `\override` can be undone by `\revert`.

The syntax for the `\revert` command is

```
\revert [context.]GrobName.property
```

For example,

```
\relative {
  c''4
  \override Voice.Stem.thickness = #3.0
  c4 c
  \revert Voice.Stem.thickness
  c4
}
```



The effects of `\override` and `\revert` apply to all grobs in the affected context from the current time forward:

```
<<
  \relative {
    e''4
    \override Staff.Stem.thickness = #3.0
    e4 e e
  } \\\
  \relative {
```

```

c''4 c c
\revert Staff.Stem.thickness
c4
}
>>

```



`\once` can be used with `\override` or `\revert` to affect only the current time step:

```

<<
\relative c {
  \override Stem.thickness = #3.0
  e''4 e e e
} \\\
\relative {
  c''4
  \once \override Stem.thickness = #3.0
  c4 c c
}
>>

```



## Vedi anche

Guida al funzionamento interno: Sezione “Backend” in *Guida al Funzionamento Interno*

### 5.3.4 Il comando `\tweak`

Changing grob properties with `\override` causes the changes to apply to all of the given grobs in the context at the moment the change applies. Sometimes, however, it is desirable to have changes apply to just one grob, rather than to all grobs in the affected context. This is accomplished with the `\tweak` command, which has the following syntax:

```
\tweak [layout-object.]grob-property value
```

Specifying *layout-object* is optional. The `\tweak` command applies to the music object that immediately follows *value* in the music stream.

For an introduction to the syntax and uses of the `tweak` command see Sezione “Metodi di modifica” in *Manuale di Apprendimento*.

When several similar items are placed at the same musical moment, the `\override` command cannot be used to modify just one of them – this is where the `\tweak` command must be used. Items which may appear more than once at the same musical moment include the following:

- note heads of notes inside a chord
- articulation signs on a single note
- ties between notes in a chord
- tuplet brackets starting at the same time

In this example, the color of one note head and the type of another note head are modified within a single chord:

```
< c''
  \tweak color #red
  d''
  g''
  \tweak duration-log #1
  a''
> 4
```



`\tweak` can be used to modify slurs:

```
\relative { c' - \tweak thickness #5 ( d e f ) }
```



For the `\tweak` command to work, it must remain immediately adjacent to the object to which it is to apply after the input file has been converted to a music stream. Tweaking a whole chord does not do anything since its music event only acts as a container, and all layout objects are created from events inside of the `EventChord`:

```
\tweak color #red c''4
\tweak color #red <c'' e''>4
<\tweak color #red c'' e''>4
```



The simple `\tweak` command cannot be used to modify any object that is not directly created from the input. In particular it will not affect stems, automatic beams or accidentals, since these are generated later by `NoteHead` layout objects rather than by music elements in the input stream.

Such indirectly created layout objects can be tweaked using the form of the `\tweak` command in which the grob name is specified explicitly:

```
\tweak Stem.color #red
\tweak Beam.color #green c''8 e''
<c'' e'' \tweak Accidental.font-size #-3 ges''>4
```



`\tweak` cannot be used to modify clefs or time signatures, since these become separated from any preceding `\tweak` command in the input stream by the automatic insertion of extra elements required to specify the context.

Several `\tweak` commands may be placed before a notational element – all affect it:

```
c'
-\tweak style #'dashed-line
-\tweak dash-fraction #0.2
-\tweak thickness #3
-\tweak color #red
\glissando
f''
```



The music stream which is generated from a section of an input file, including any automatically inserted elements, may be examined, see Sezione “Displaying music expressions” in *Estendere*. This may be helpful in determining what may be modified by a `\tweak` command, or in determining how to adjust the input to make a `\tweak` apply.

## Vedi anche

Manuale di apprendimento: Sezione “Metodi di modifica” in *Manuale di Apprendimento*.

Estendere LilyPond: Sezione “Displaying music expressions” in *Estendere*.

## Problemi noti e avvertimenti

The `\tweak` command cannot be used to modify the control points of just one of several ties in a chord, other than the first one encountered in the input file.

### 5.3.5 `\set` vs. `\override`

Both `\set` and `\override` manipulate properties associated with contexts. In either case, properties heed the hierarchy of contexts: properties not set in a context itself show the values of the respective parent context.

Values and lifetime of context properties are dynamic and only available when music is being interpreted, ‘iterated’. At the time of context creation, properties are initialized from the corresponding context definition and possible context modifications. Afterwards, changes are achieved with property-setting commands in the music itself.

Now grob definitions are a special category of context properties. Since their structure, bookkeeping and use is different from ordinary context properties, they are accessed with a different set of commands, and treated separately in the documentation.

As opposed to plain context properties, grob definitions are subdivided into grob properties. A “grob” (graphical object) is usually created by an engraver at the time of interpreting a music expression and receives its initial properties from the current grob definition of the engraver’s context. The engraver (or other ‘backend’ parts of LilyPond) may subsequently add or change properties to the grob, but that does not affect the context’s grob definition.

What we call ‘grob properties’ in the context of user-level tweaking are actually the properties of a context’s grob definition. In contrast to ordinary context properties, grob definitions have the bookkeeping required to keep track of its parts, the individual grob properties (and even subproperties of them) separately so that it is possible to define those parts in different contexts and have the overall grob definition at the time of grob creation be assembled from pieces provided in different contexts among the current context and its parents.

Grob definitions are manipulated using `\override` and `\revert` and have a name starting with a capital letter (like ‘NoteHead’) whereas ordinary context properties are manipulated using `\set` and `\unset` and are named starting with a lowercase letter.



The special commands `\tweak` and `\overrideProperty` change grob properties bypassing context properties completely. Instead they catch grobs as they are being created and then directly set properties on them when they originate from a tweaked music event or are of a particular kind, respectively.

### 5.3.6 Modifying alists

Some user-configurable properties are internally represented as *alists* (association lists), which store pairs of *keys* and *values*. The structure of an alist is:

```
'((key1 . value1)
  (key2 . value2)
  (key3 . value3)
  ...)
```

If an alist is a grob property or `\paper` variable, its keys can be modified individually without affecting other keys.

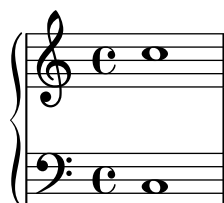
For example, to reduce the space between adjacent staves in a staff-group, use the `staff-staff-spacing` property of the `StaffGrouper` grob. The property is an alist with four keys: `basic-distance`, `minimum-distance`, `padding`, and `stretchability`. The standard settings for this property are listed in the “Backend” section of the Guida al funzionamento interno (see Sezione “StaffGrouper” in *Guida al Funzionamento Interno*):

```
'((basic-distance . 9)
  (minimum-distance . 7)
  (padding . 1)
  (stretchability . 5))
```

One way to bring the staves closer together is by reducing the value of the `basic-distance` key (9) to match the value of `minimum-distance` (7). To modify a single key individually, use a *nested declaration*:

```
% default space between staves
\new PianoStaff <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass c1 }
>>

% reduced space between staves
\new PianoStaff \with {
  % this is the nested declaration
  \override StaffGrouper.staff-staff-spacing.basic-distance = #7
} <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass c1 }
>>
```





Using a nested declaration will update the specified key (such as `basic-distance` in the above example) without altering any other keys already set for the same property.

Now suppose we want the staves to be as close as possible without overlapping. The simplest way to do this is to set all four alist keys to zero. However, it is not necessary to enter four nested declarations, one for each key. Instead, the property can be completely re-defined with one declaration, as an alist:

```
\new PianoStaff \with {
  \override StaffGrouper.staff-staff-spacing =
    #'((basic-distance . 0)
      (minimum-distance . 0)
      (padding . 0)
      (stretchability . 0))
} <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass c1 }
>>
```



Note that any keys not explicitly listed in the alist definition will be reset to their *default-when-unset* values. In the case of `staff-staff-spacing`, any unset key-values would be reset to zero (except `stretchability`, which takes the value of `basic-distance` when unset). Thus the following two declarations are equivalent:

```
\override StaffGrouper.staff-staff-spacing =
  #'((basic-distance . 7))

\override StaffGrouper.staff-staff-spacing =
  #'((basic-distance . 7)
    (minimum-distance . 0)
    (padding . 0)
    (stretchability . 7))
```

One (possibly unintended) consequence of this is the removal of any standard settings that are set in an initialization file and loaded each time an input file is compiled. In the above example, the standard settings for `padding` and `minimum-distance` (defined in `scm/define-grobs.scm`) are reset to their default-when-unset values (zero for both keys). Defining a property or variable as an alist (of any size) will always reset all unset key-values to their default-when-unset values. Unless this is the intended result, it is safer to update key-values individually with a nested declaration.

**Nota:** Nested declarations will not work for context property alists (such as `beamExceptions`, `keyAlterations`, `timeSignatureSettings`, etc.). These properties can only be modified by completely re-defining them as alists.

## 5.4 Proprietà e concetti utili

### 5.4.1 Input modes

The way in which the notation contained within an input file is interpreted is determined by the current input mode.

#### *Chord mode*

This is activated with the `\chordmode` command, and causes input to be interpreted with the syntax of chord notation, see Sezione 2.7 [Chord notation], pagina 411. Chords are rendered as notes on a staff.

Chord mode is also activated with the `\chords` command. This also creates a new **ChordNames** context and causes the following input to be interpreted with the syntax of chord notation and rendered as chord names in the **ChordNames** context, see [Printing chord names], pagina 416.

#### *Drum mode*

This is activated with the `\drummode` command, and causes input to be interpreted with the syntax of drum notation, see [Basic percussion notation], pagina 388.

Drum mode is also activated with the `\drums` command. This also creates a new **DrumStaff** context and causes the following input to be interpreted with the syntax of drum notation and rendered as drum symbols on a drum staff, see [Basic percussion notation], pagina 388.

#### *Figure mode*

This is activated with the `\figuremode` command, and causes input to be interpreted with the syntax of figured bass, see [Entering figured bass], pagina 425.

Figure mode is also activated with the `\figures` command. This also creates a new **FiguredBass** context and causes the following input to be interpreted with the figured bass syntax and rendered as figured bass symbols in the **FiguredBass** context, see [Introduction to figured bass], pagina 424.

#### *Fret and tab modes*

There are no special input modes for entering fret and tab symbols.

To create tab diagrams, enter notes or chords in note mode and render them in a **TabStaff** context, see [Default tablatures], pagina 341.

To create fret diagrams above a staff, you have two choices. You can either use the **FretBoards** context (see [Automatic fret diagrams], pagina 379, or you can enter them as a markup above the notes using the `\fret-diagram` command (see [Fret diagram markups], pagina 359).

#### *Lyrics mode*

This is activated with the `\lyricmode` command, and causes input to be interpreted as lyric syllables with optional durations and associated lyric modifiers, see <undefined> [Vocal music], pagina <undefined>.

Lyric mode is also activated with the `\addlyrics` command. This also creates a new **Lyrics** context and an implicit `\lyricsto` command which associates the following lyrics with the preceding music.

#### *Markup mode*

This is activated with the `\markup` command, and causes input to be interpreted with the syntax of markup, see <undefined> [Text markup commands], pagina <undefined>.

## Note mode

This is the default mode or it may be activated with the `\notemode` command. Input is interpreted as pitches, durations, markup, etc and typeset as musical notation on a staff.

It is not normally necessary to specify note mode explicitly, but it may be useful to do so in certain situations, for example if you are in lyric mode, chord mode or any other mode and want to insert something that only can be done with note mode syntax.

### 5.4.2 Direction and placement

In typesetting music the direction and placement of many items is a matter of choice. For example, the stems of notes can be directed up or down; lyrics, dynamics, and other expressive marks may be placed above or below the staff; text may be aligned left, right or center; etc. Most of these choices may be left to be determined automatically by LilyPond, but in some cases it may be desirable to force a particular direction or placement.

### Articulation direction indicators

By default some directions are always up or always down (e.g. dynamics or fermata), while other things can alternate between up or down based on the stem direction (like slurs or accents).

The default action may be overridden by prefixing the articulation by a *direction indicator*. Three direction indicators are available: `^` (meaning “up”), `_` (meaning “down”) and `-` (meaning “use default direction”). The direction indicator can usually be omitted, in which case `-` is assumed, but a direction indicator is **always** required before

- `\tweak` commands
- `\markup` commands
- `\tag` commands
- string markups, e.g. `-"string"`
- fingering instructions, e.g. `-1`
- articulation shortcuts, e.g. `-. , -> , --`

Direction indicators affect only the next note:

```
\relative {
  c''2( c)
  c2_( c)
  c2( c)
  c2^( c)
}
```



### The direction property

The position or direction of many layout objects is controlled by the `direction` property.

The value of the `direction` property may be set to `1`, meaning “up” or “above”, or to `-1`, meaning “down” or “below”. The symbols `UP` and `DOWN` may be used instead of `1` and `-1` respectively. The default direction may be specified by setting `direction` to `0` or `CENTER`. Alternatively, in many cases predefined commands exist to specify the direction. These are of the form

```
\xxxUp, \xxxDown or \xxxNeutral
```

where `\xxxNeutral` means “use the default” direction. See Sezione “Oggetti interni al rigo” in *Manuale di Apprendimento*.

In a few cases, arpeggio for example, the value of the `direction` property can specify whether the object is to be placed to the right or left of the parent. In this case `-1` or `LEFT` means “to the left” and `1` or `RIGHT` means “to the right”. `0` or `CENTER` means “use the default” direction.

These indications affect all notes until they are canceled.

```
\relative {
  c''2( c)
  \slurDown
  c2( c)
  c2( c)
  \slurNeutral
  c2( c)
}
```



In polyphonic music, it is generally better to specify an explicit `voice` than change an object’s direction. For more information. See [\[Multiple voices\]](#), pagina [\[Multiple voices\]](#).

## Vedi anche

Manuale di apprendimento: Sezione “Oggetti interni al rigo” in *Manuale di Apprendimento*.

Guida alla notazione: [\[Multiple voices\]](#), pagina [\[Multiple voices\]](#).

### 5.4.3 Distances and measurements

Distances in LilyPond are of two types: absolute and scaled.

Absolute distances are used for specifying margins, indents, and other page layout details, and are by default specified in millimeters. Distances may be specified in other units by following the quantity by `\mm`, `\cm`, `\in` (inches), or `\pt` (points, 1/72.27 of an inch). Page layout distances can also be specified in scalable units (see the following paragraph) by appending `\staff-space` to the quantity. Page layout is described in detail in [\[Page layout\]](#), pagina [\[Page layout\]](#).

Scaled distances are always specified in units of the staff-space or, rarely, the half staff-space. The staff-space is the distance between two adjacent staff lines. The default value can be changed globally by setting the global staff size, or it can be overridden locally by changing the `staff-space` property of `StaffSymbol`. Scaled distances automatically scale with any change to the either the global staff size or the `staff-space` property of `StaffSymbol`, but fonts scale automatically only with changes to the global staff size. The global staff size thus enables the overall size of a rendered score to be easily varied. For the methods of setting the global staff size see [\[Setting the staff size\]](#), pagina [\[Setting the staff size\]](#).

If just a section of a score needs to be rendered to a different scale, for example an ossia section or a footnote, the global staff size cannot simply be changed as this would affect the entire score. In such cases the change in size is made by overriding both the `staff-space` property of `StaffSymbol` and the size of the fonts. A Scheme function, `magstep`, is available to convert from a font size change to the equivalent change in `staff-space`. For an explanation and an example of its use, see Sezione “Lunghezza e spessore degli oggetti” in *Manuale di Apprendimento*.

## Vedi anche

Manuale di apprendimento: Sezione “Lunghezza e spessore degli oggetti” in *Manuale di Apprendimento*.

Guida alla notazione: `<undefined>` [Page layout], pagina `<undefined>`, `<undefined>` [Setting the staff size], pagina `<undefined>`.

### 5.4.4 Dimensions

The dimensions of a graphical object specify the positions of the left and right edges and the bottom and top edges of the objects’ bounding box as distances from the objects’ reference point in units of staff-spaces. These positions are usually coded as two Scheme pairs. For example, the text markup command `\with-dimensions` takes three arguments, the first two of which are a Scheme pair giving the left and right edge positions and a Scheme pair giving the bottom and top edge positions:

```
\with-dimensions #'(-5 . 10) #'(-3 . 15) arg
```

This specifies a bounding box for `arg` with its left edge at -5, its right edge at 10, its bottom edge at -3 and its top edge at 15, all measured from the objects’ reference point in units of staff-spaces.

## Vedi anche

Guida alla notazione: Sezione 5.4.3 [Distances and measurements], pagina 612.

### 5.4.5 Staff symbol properties

The vertical position of staff lines and the number of staff lines can be defined at the same time. As the following example shows, note positions are not influenced by the staff line positions.

**Nota:** The `'line-positions` property overrides the `'line-count` property. The number of staff lines is implicitly defined by the number of elements in the list of values for `'line-positions`.

```
\new Staff \with {
  \override StaffSymbol.line-positions = #'(7 3 0 -4 -6 -7)
}
\relative { a4 e' f b | d1 }
```



The width of a staff can be modified. The units are staff spaces. The spacing of objects inside the staff is not affected by this setting.

```
\new Staff \with {
  \override StaffSymbol.width = #23
}
\relative { a4 e' f b | d1 }
```



### 5.4.6 Spanners

Many objects of musical notation extend over several notes or even several bars. Examples are slurs, beams, tuplet brackets, volta repeat brackets, crescendi, trills, and glissandi. Such objects are collectively called “spanners”, and have special properties to control their appearance and behaviour. Some of these properties are common to all spanners; others are restricted to a sub-set of the spanners.

All spanners support the `spanner-interface`. A few, essentially those that draw a straight line between the two objects, support in addition the `line-spanner-interface`.

#### Using the `spanner-interface`

This interface provides two properties that apply to several spanners.

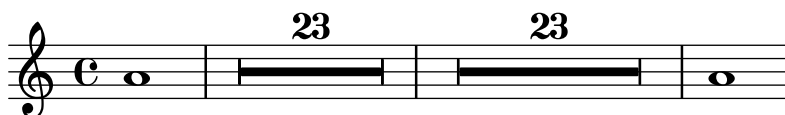
##### *The minimum-length property*

The minimum length of the spanner is specified by the `minimum-length` property. Increasing this usually has the necessary effect of increasing the spacing of the notes between the two end points. However, this override has no effect on many spanners, as their length is determined by other considerations. A few examples where it is effective are shown below.

```
a'~ a'
a'
% increase the length of the tie
-\tweak minimum-length #5
~ a'
```



```
\relative \compressMMRests {
  a'1
  R1*23
  % increase the length of the rest bar
  \once \override MultiMeasureRest.minimum-length = #20
  R1*23
  a1
}
```



```
\relative {
  a' \< a a a \!
  % increase the length of the hairpin
  \override Hairpin.minimum-length = #20
  a \< a a a \!
}
```



This override can also be used to increase the length of slurs and phrasing slurs:

```
\relative {
  a' ( g)
  a
  -\tweak minimum-length #5
  ( g)

  a\ ( g\ )
  a
  -\tweak minimum-length #5
  \ ( g\ )
}
```



For some layout objects, the `minimum-length` property becomes effective only if the `set-spacing-rods` procedure is called explicitly. To do this, the `springs-and-rods` property should be set to `ly:spanner::set-spacing-rods`. For example, the minimum length of a glissando has no effect unless the `springs-and-rods` property is set:

```
% default
e' \glissando c''

% not effective alone
\once \override Glissando.minimum-length = #20
e' \glissando c''

% effective only when both overrides are present
\once \override Glissando.minimum-length = #20
\once \override Glissando.springs-and-rods = #ly:spanner::set-spacing-rods
e' \glissando c''
```



The same is true of the `Beam` object:

```
% not effective alone
\once \override Beam.minimum-length = #20
e'8 e' e' e'

% effective only when both overrides are present
\once \override Beam.minimum-length = #20
\once \override Beam.springs-and-rods = #ly:spanner::set-spacing-rods
e'8 e' e' e'
```

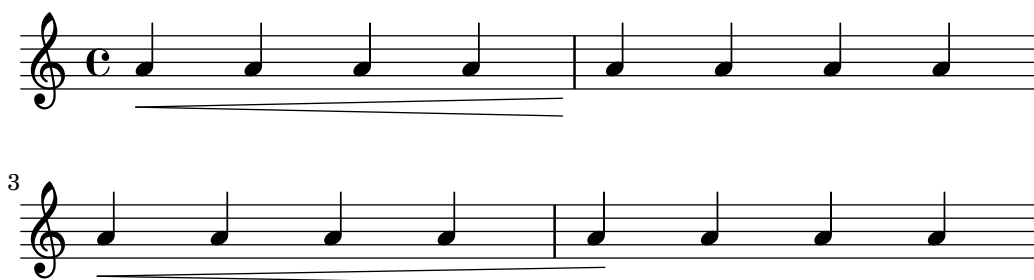




### *The to-barline property*

The second useful property of the `spanner-interface` is `to-barline`. By default this is true, causing hairpins and other spanners which are terminated on the first note of a measure to end instead on the immediately preceding bar line. If set to false, the spanner will extend beyond the bar line and end on the note itself:

```
\relative {
  a' \< a a a a \! a a a \break
  \override Hairpin.to-barline = ##f
  a \< a a a a \! a a a
}
```



This property is not effective for all spanners. For example, setting it to `#t` has no effect on slurs or phrasing slurs or on other spanners for which terminating on the bar line would not be meaningful.

### Using the line-spanner-interface

Objects which support the `line-spanner-interface` include

- `DynamicTextSpanner`
- `Glissando`
- `TextSpanner`
- `TrillSpanner`
- `VoiceFollower`

The routine responsible for drawing the stencils for these spanners is `ly:line-spanner::print`. This routine determines the exact location of the two end points and draws a line between them, in the style requested. The locations of the two end points of the spanner are computed on-the-fly, but it is possible to override their Y-coordinates. The properties which need to be specified are nested two levels down within the property hierarchy, but the syntax of the `\override` command is quite simple:

```
e''2 \glissando b'
\once \override Glissando.bound-details.left.Y = #3
\once \override Glissando.bound-details.right.Y = #-2
e''2 \glissando b'
```



The units for the Y property are `staff-spaces`, with the center line of the staff being the zero point. For the glissando, this is the value for Y at the X-coordinate corresponding to the center point of each note head, if the line is imagined to be extended to there.

If `Y` is not set, the value is computed from the vertical position of the corresponding attachment point of the spanner.

In case of a line break, the values for the end points are specified by the `left-broken` and `right-broken` sub-lists of `bound-details`. For example:

```
\override Glissando.breakable = ##t
\override Glissando.bound-details.right-broken.Y = #-3
c''1 \glissando \break
f''1
```



A number of further properties of the `left` and `right` sub-lists of the `bound-details` property may be modified in the same way as `Y`:

**Y** This sets the Y-coordinate of the end point, in `staff-spaces` offset from the staff center line. By default, it is the center of the bound object, so a glissando points to the vertical center of the note head.

For horizontal spanners, such as text spanners and trill spanners, it is hardcoded to 0.

**attach-dir**

This determines where the line starts and ends in the X-direction, relative to the bound object. So, a value of `-1` (or `LEFT`) makes the line start/end at the left side of the note head it is attached to.

**X** This is the absolute X-coordinate of the end point. It is usually computed on the fly, and overriding it has little useful effect.

**stencil** Line spanners may have symbols at the beginning or end, which is contained in this sub-property. This is for internal use; it is recommended that `text` be used instead.

**text** This is a markup that is evaluated to yield the stencil. It is used to put *cresc.*, *tr* and other text on horizontal spanners.

```
\override TextSpanner.bound-details.left.text
= \markup { \small \bold Slower }
\relative { c''2\startTextSpan b c a\stopTextSpan }
```



**stencil-align-dir-y**

**stencil-offset**

Without setting one of these, the stencil is simply put at the end-point, centered on the line, as defined by the `X` and `Y` sub-properties. Setting either `stencil-align-dir-y` or `stencil-offset` will move the symbol at the edge vertically relative to the end point of the line:

```
\override TextSpanner.bound-details.left.stencil-align-dir-y = #-2
```

```

\override TextSpanner.bound-details.right.stencil-align-dir-y = #UP

\override TextSpanner.bound-details.left.text = #"ggg"
\override TextSpanner.bound-details.right.text = #"hhh"

\relative { c'4^\startTextSpan c c c \stopTextSpan }

```



Note that negative values move the text *up*, contrary to the effect that might be expected, as a value of -1 or DOWN means align the *bottom* edge of the text with the spanner line. A value of 1 or UP aligns the top edge of the text with the spanner line.

- arrow** Setting this sub-property to **#t** produces an arrowhead at the end of the line.
- padding** This sub-property controls the space between the specified end point of the line and the actual end. Without padding, a glissando would start and end in the center of each note head.

The music function `\endSpanners` terminates the spanner which starts on the immediately following note prematurely. It is terminated after exactly one note, or at the following bar line if `to-barline` is true and a bar line occurs before the next note.

```

\relative c' {
  \endSpanners
  c2 \startTextSpan c2 c2
  \endSpanners
  c2 \< c2 c2
}

```



When using `\endSpanners` it is not necessary to close `\startTextSpan` with `\stopTextSpan`, nor is it necessary to close hairpins with `\!`.

## Vedi anche

Guida al funzionamento interno: Sezione “TextSpanner” in *Guida al Funzionamento Interno*, Sezione “Glissando” in *Guida al Funzionamento Interno*, Sezione “VoiceFollower” in *Guida al Funzionamento Interno*, Sezione “TrillSpanner” in *Guida al Funzionamento Interno*, Sezione “line-spanner-interface” in *Guida al Funzionamento Interno*.

### 5.4.7 Visibility of objects

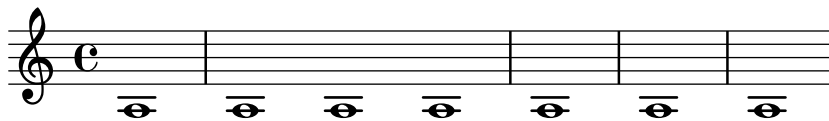
There are four main ways in which the visibility of layout objects can be controlled: their stencil can be removed, they can be made transparent, they can be colored white, or their `break-visibility` property can be overridden. The first three apply to all layout objects; the last to just a few – the *breakable* objects. The Manuale di apprendimento introduces these four techniques, see Sezione “Visibilità e colore degli oggetti” in *Manuale di Apprendimento*.

There are also a few other techniques which are specific to certain layout objects. These are covered under Special considerations.

## Removing the stencil

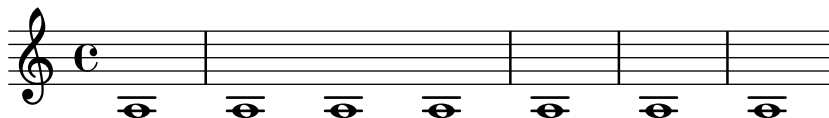
Every layout object has a `stencil` property. By default this is set to the specific function which draws that object. If this property is overridden to `#f` no function will be called and the object will not be drawn. The default action can be recovered with `\revert`.

```
a1 a
\override Score.BarLine.stencil = ##f
a a
\revert Score.BarLine.stencil
a a a
```



This rather common operation has a shortcut `\omit`:

```
a1 a
\omit Score.BarLine
a a
\undo \omit Score.BarLine
a a a
```



## Making objects transparent

Every layout object has a `transparent` property which by default is set to `#f`. If set to `#t` the object still occupies space but is made invisible.

```
a'4 a'
\once \override NoteHead.transparent = ##t
a' a'
```



This rather common operation has a shortcut `\hide`:

```
a'4 a'
\once \hide NoteHead
a' a'
```



## Painting objects white

Every layout object has a color property which by default is set to **black**. If this is overridden to **white** the object will be indistinguishable from the white background. However, if the object crosses other objects the color of the crossing points will be determined by the order in which they are drawn, and this may leave a ghostly image of the white object, as shown here:

```
\override Staff.Clef.color = #white
a'1
```



This may be avoided by changing the order of printing the objects. All layout objects have a **layer** property which should be set to an integer. Objects with the lowest value of **layer** are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a **layer** value of 1, although a few objects, including **StaffSymbol** and **BarLine**, are assigned a value of 0. The order of printing objects with the same value of **layer** is indeterminate.

In the example above the white clef, with a default **layer** value of 1, is drawn after the staff lines (default **layer** value 0), so overwriting them. To change this, the **Clef** object must be given in a lower value of **layer**, say -1, so that it is drawn earlier:

```
\override Staff.Clef.color = #white
\override Staff.Clef.layer = #-1
a'1
```



## Using break-visibility

Most layout objects are printed only once, but some like bar lines, clefs, time signatures and key signatures, may need to be printed twice when a line break occurs – once at the end of the line and again at the start of the next line. Such objects are called *breakable*, and have a property, the **break-visibility** property to control their visibility at the three positions in which they may appear – at the start of a line, within a line if they are changed, and at the end of a line if a change takes place there.

For example, the time signature by default will be printed at the start of the first line, but nowhere else unless it changes, when it will be printed at the point at which the change occurs. If this change occurs at the end of a line the new time signature will be printed at the start of the next line and a cautionary time signature will be printed at the end of the previous line as well.

This behaviour is controlled by the **break-visibility** property, which is explained in Sezione “Visibilità e colore degli oggetti” in *Manuale di Apprendimento*. This property takes a vector of three booleans which, in order, determine whether the object is printed at the end of, within the body of, or at the beginning of a line. Or to be more precise, before a line break, where there is no line break, or after a line break.

Alternatively, these eight combinations may be specified by pre-defined functions, defined in `scm/output-lib.scm`, where the last three columns indicate whether the layout objects will be visible in the positions shown at the head of the columns:

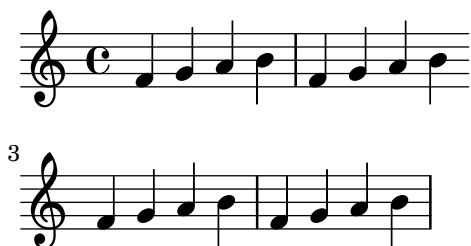
Function form	Vector form	Before break	At no break	After break
all-visible	##(t t t)	yes	yes	yes
begin-of-line-visible	##(f f t)	no	no	yes
center-visible	##(f t f)	no	yes	no
end-of-line-visible	##(t f f)	yes	no	no
begin-of-line-invisible	##(t t f)	yes	yes	no
center-invisible	##(t f t)	yes	no	yes
end-of-line-invisible	##(f t t)	no	yes	yes
all-invisible	##(f f f)	no	no	no

The default settings of `break-visibility` depend on the layout object. The following table shows all the layout objects of interest which are affected by `break-visibility` and the default setting of this property:

Layout object	Usual context	Default setting
BarLine	Score	calculated
BarNumber	Score	begin-of-line-visible
BreathingSign	Voice	begin-of-line-invisible
Clef	Staff	begin-of-line-visible
Custos	Staff	end-of-line-visible
DoublePercentRepeat	Voice	begin-of-line-invisible
KeyCancellation	Staff	begin-of-line-invisible
KeySignature	Staff	begin-of-line-visible
ClefModifier	Staff	begin-of-line-visible
RehearsalMark	Score	end-of-line-invisible
TimeSignature	Staff	all-visible

The example below shows the use of the vector form to control the visibility of bar lines:

```
\relative {
  f'4 g a b
  f4 g a b
  % Remove bar line at the end of the current line
  \once \override Score.BarLine.break-visibility = ##(f t t)
  \break
  f4 g a b
  f4 g a b
}
```



Although all three components of the vector used to override `break-visibility` must be present, not all of them are effective with every layout object, and some combinations may even give errors. The following limitations apply:

- Bar lines cannot be printed at start of line.

- A bar number cannot be printed at the start of the first line unless it is set to be different from 1.
- Clef – see below
- Double percent repeats are either all printed or all suppressed. Use `begin-of` line-invisible to print and `all-invisible` to suppress.
- Key signature – see below
- `ClefModifier` – see below

## Special considerations

### *Visibility following explicit changes*

The `break-visibility` property controls the visibility of key signatures and changes of clef only at the start of lines, i.e. after a break. It has no effect on the visibility of the key signature or clef following an explicit key change or an explicit clef change within or at the end of a line. In the following example the key signature following the explicit change to B-flat major is still visible, even though `all-invisible` is set.

```
\relative {
  \key g \major
  f'4 g a b
  % Try to remove all key signatures
  \override Staff.KeySignature.break-visibility = #all-invisible
  \key bes \major
  f4 g a b
  \break
  f4 g a b
  f4 g a b
}
```



The visibility of such explicit key signature and clef changes is controlled by the `explicitKeySignatureVisibility` and `explicitClefVisibility` properties. These are the equivalent of the `break-visibility` property and both take a vector of three booleans or the predefined functions listed above, exactly like `break-visibility`. Both are properties of the `Staff` context, not the layout objects themselves, and so they are set using the `\set` command. Both are set by default to `all-visible`. These properties control only the visibility of key signatures and clefs resulting from explicit changes and do not affect key signatures and clefs at the beginning of lines; `break-visibility` must still be overridden in the appropriate object to remove these.

```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \override Staff.KeySignature.break-visibility = #all-invisible
```

```
\key bes \major
f4 g a b \break
f4 g a b
f4 g a b
}
```



### *Visibility of cancelling accidentals*

To remove the cancelling accidentals printed at an explicit key change, set the Staff context property `printKeyCancellation` to `#f`:

```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \set Staff.printKeyCancellation = ##f
  \override Staff.KeySignature.break-visibility = #all-invisible
  \key bes \major
  f4 g a b \break
  f4 g a b
  f4 g a b
}
```



With these overrides only the accidentals before the notes remain to indicate the change of key.

Note that when changing the key to C major or A minor the cancelling accidentals would be the *only* indication of the key change. In this case setting `printKeyCancellation` to `#f` has no effect:

```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \set Staff.printKeyCancellation = ##f
  \key c \major
  f4 g a b \break
  f4 g a b
}
```



```
f4 g a b
}
```



To suppress the cancelling accidentals even when the key is changed to C major or A minor, override the visibility of the `KeyCancellation` grob instead:

```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \override Staff.KeyCancellation.break-visibility = #all-invisible
  \key c \major
  f4 g a b \break
  f4 g a b
  f4 g a b
}
```



### *Automatic bars*

As a special case, the printing of bar lines can also be turned off by setting the `automaticBars` property in the `Score` context. If set to `#f`, bar lines will not be printed automatically; they must be explicitly created with a `\bar` command. Unlike the `\cadenzaOn` predefined command, measures are still counted. Bar generation will resume according to that count if this property is later set to `#t`. When set to `#f`, line breaks can occur only at explicit `\bar` commands.

### *Transposed clefs*

The small transposition symbol on transposed clefs is produced by the `ClefModifier` layout object. Its visibility is automatically inherited from the `Clef` object, so it is not necessary to apply any required `break-visibility` overrides to the `ClefModifier` layout objects to suppress transposition symbols for invisible clefs.

For explicit clef changes, the `explicitClefVisibility` property controls both the clef symbol and any transposition symbol associated with it.

### **Vedi anche**

Manuale di apprendimento: Sezione “Visibilità e colore degli oggetti” in *Manuale di Apprendimento*.

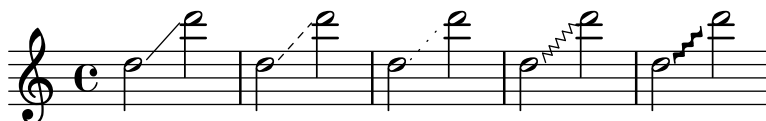
### 5.4.8 Line styles

Some performance indications, e.g., *rallentando* and *accelerando* and *trills* are written as text and are extended over many measures with lines, sometimes dotted or wavy.

These all use the same routines as the glissando for drawing the texts and the lines, and tuning their behavior is therefore also done in the same way. It is done with a spanner, and the routine responsible for drawing the spanners is `ly:line-spanner::print`. This routine determines the exact location of the two *span points* and draws a line between them, in the style requested.

Here is an example showing the different line styles available, and how to tune them.

```
\relative {
  d''2 \glissando d'2
  \once \override Glissando.style = #'dashed-line
  d,2 \glissando d'2
  \override Glissando.style = #'dotted-line
  d,2 \glissando d'2
  \override Glissando.style = #'zigzag
  d,2 \glissando d'2
  \override Glissando.style = #'trill
  d,2 \glissando d'2
}
```



The locations of the end-points of the spanner are computed on-the-fly for every graphic object, but it is possible to override these:

```
\relative {
  e''2 \glissando f
  \once \override Glissando.bound-details.right.Y = #-2
  e2 \glissando f
}
```



The value for Y is set to -2 for the right end point. The left side may be similarly adjusted by specifying `left` instead of `right`.

If Y is not set, the value is computed from the vertical position of the left and right attachment points of the spanner.

Other adjustments of spanners are possible, for details, see Sezione 5.4.6 [Spanners], pagina 614.

### 5.4.9 Rotating objects

Both layout objects and elements of markup text can be rotated by any angle about any point, but the method of doing so differs.

## Rotating layout objects

All layout objects which support the `grob-interface` can be rotated by setting their `rotation` property. This takes a list of three items: the angle of rotation counter-clockwise, and the x and y coordinates of the point relative to the object's reference point about which the rotation is to be performed. The angle of rotation is specified in degrees and the coordinates in staff-spaces.

The angle of rotation and the coordinates of the rotation point must be determined by trial and error.

There are only a few situations where the rotation of layout objects is useful; the following example shows one situation where they may be:

```
g4\< e' d'' f''\!
\override Hairpin.rotation = #'(20 -1 0)
g4\< e' d'' f''\!
```



## Rotating markup

All markup text can be rotated to lie at any angle by prefixing it with the `\rotate` command. The command takes two arguments: the angle of rotation in degrees counter-clockwise and the text to be rotated. The extents of the text are not rotated: they take their values from the extremes of the x and y coordinates of the rotated text. In the following example the `outside-staff-priority` property for text is set to `##f` to disable the automatic collision avoidance, which would push some of the text too high.

```
\override TextScript.outside-staff-priority = ##f
g4^\markup { \rotate #30 "a G" }
b^\markup { \rotate #30 "a B" }
des'^\markup { \rotate #30 "a D-Flat" }
fis'^\markup { \rotate #30 "an F-Sharp" }
```



## 5.5 Ritocchi avanzati

This section discusses various approaches to fine tuning the appearance of the printed score.

### Vedi anche

Manuale di apprendimento: Sezione “Modifica dell’output” in *Manuale di Apprendimento*, Sezione “Altre fonti di informazione” in *Manuale di Apprendimento*.

Guida alla notazione: [\[Explaining the Internals Reference\]](#), pagina [\[Modifying properties\]](#).

Estendere LilyPond: Sezione “Interfaces for programmers” in *Estendere*.

File installati: `scm/define-grobs.scm`.

Frammenti: Sezione “Tweaks and overrides” in *Frammenti di codice*.

Guida al funzionamento interno: Sezione “All layout objects” in *Guida al Funzionamento Interno*.

### 5.5.1 Aligning objects

Graphical objects which support the `self-alignment-interface` and/or the `side-position-interface` can be aligned to a previously placed object in a variety of ways. For a list of these objects, see Sezione “self-alignment-interface” in *Guida al Funzionamento Interno* and Sezione “side-position-interface” in *Guida al Funzionamento Interno*.

All graphical objects have a reference point, a horizontal extent and a vertical extent. The horizontal extent is a pair of numbers giving the displacements from the reference point of the left and right edges, displacements to the left being negative. The vertical extent is a pair of numbers giving the displacement from the reference point to the bottom and top edges, displacements down being negative.

An object’s position on a staff is given by the values of the `X-offset` and `Y-offset` properties. The value of `X-offset` gives the displacement from the X coordinate of the reference point of the parent object, and the value of `Y-offset` gives the displacement from the center line of the staff. The values of `X-offset` and `Y-offset` may be set directly or may be set to be calculated by procedures in order to achieve alignment with the parent object.

**Nota:** Many objects have special positioning considerations which cause any setting of `X-offset` or `Y-offset` to be ignored or modified, even though the object supports the `self-alignment-interface`. Overriding the `X-offset` or `Y-offset` properties to a fixed value causes the respective `self-alignment` property to be disregarded.

For example, an accidental can be repositioned vertically by setting `Y-offset` but any changes to `X-offset` have no effect.

Rehearsal marks may be aligned with breakable objects such as bar lines, clef symbols, time signature symbols and key signatures. There are special properties to be found in the `break-aligned-interface` for positioning rehearsal marks on such objects.

## Vedi anche

Guida alla notazione: [Using the break-alignable-interface], pagina 629.

Estendere LilyPond: Sezione “Callback functions” in *Estendere*.

## Setting X-offset and Y-offset directly

Numerical values may be given to the `X-offset` and `Y-offset` properties of many objects. The following example shows three notes with the default fingering position and the positions with `X-offset` and `Y-offset` modified.

```
a'-3
a'
-\tweak X-offset #0
-\tweak Y-offset #0
-3
a'
-\tweak X-offset #-1
-\tweak Y-offset #1
-3
```



## Using the side-position-interface

An object which supports the `side-position-interface` can be placed next to its parent object so that the specified edges of the two objects touch. The object may be placed above, below, to the right or to the left of the parent. The parent cannot be specified; it is determined by the order of elements in the input stream. Most objects have the associated note head as their parent.

The values of the `side-axis` and `direction` properties determine where the object is to be placed, as follows:

<code>side-axis</code> property	<code>direction</code> property	Placement
0	-1	left
0	1	right
1	-1	below
1	1	above

When `side-axis` is 0, `X-offset` should be set to the procedure `ly:side-position-interface::x-aligned-side`. This procedure will return the correct value of `X-offset` to place the object to the left or right side of the parent according to value of `direction`.

When `side-axis` is 1, `Y-offset` should be set to the procedure `ly:side-position-interface::y-aligned-side`. This procedure will return the correct value of `Y-offset` to place the object to the top or bottom of the parent according to value of `direction`.

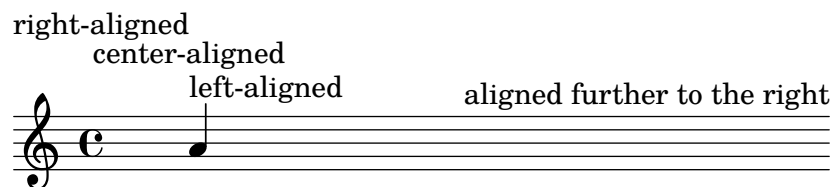
## Using the self-alignment-interface

### *Self-aligning objects horizontally*

The horizontal alignment of an object which supports the `self-alignment-interface` is controlled by the value of the `self-alignment-X` property, provided the object's `X-offset` property is set to `ly:self-alignment-interface::x-aligned-on-self`. `self-alignment-X` may be given any real value, in units of half the total X extent of the object. Negative values move the object to the right, positive to the left. A value of 0 centers the object on the reference point of its parent, a value of -1 aligns the left edge of the object on the reference point of its parent, and a value of 1 aligns the right edge of the object on the reference point of its parent. The symbols `LEFT`, `CENTER`, and `RIGHT` may be used instead of the values -1, 0, and 1, respectively.

Normally the `\override` command would be used to modify the value of `self-alignment-X`, but the `\tweak` command can be used to separately align several annotations on a single note:

```
a'
-\tweak self-alignment-X #-1
^"left-aligned"
-\tweak self-alignment-X #0
^"center-aligned"
-\tweak self-alignment-X #RIGHT
^"right-aligned"
-\tweak self-alignment-X #-2.5
^"aligned further to the right"
```



### *Self-aligning objects vertically*

Objects may be aligned vertically in an analogous way to aligning them horizontally if the `Y-offset` property is set to `ly:self-alignment-interface::y-aligned-on-self`. However, other mechanisms are often involved in vertical alignment: the value of `Y-offset` is just one variable taken into account. This may make adjusting the value of some objects tricky. The units are just half the vertical extent of the object, which is usually quite small, so quite large numbers may be required. A value of `-1` aligns the lower edge of the object with the reference point of the parent object, a value of `0` aligns the center of the object with the reference point of the parent, and a value of `1` aligns the top edge of the object with the reference point of the parent. The symbols `DOWN`, `CENTER`, and `UP` may be substituted for `-1`, `0`, and `1`, respectively.

### *Self-aligning objects in both directions*

By setting both `X-offset` and `Y-offset`, an object may be aligned in both directions simultaneously.

The following example shows how to adjust a fingering mark so that it nestles close to the note head.

```
a'
-\tweak self-alignment-X #0.5 % move horizontally left
-\tweak Y-offset #ly:self-alignment-interface::y-aligned-on-self
-\tweak self-alignment-Y #-1 % move vertically up
-3 % third finger
```



### Using the break-alignable-interface

Rehearsal marks and bar numbers may be aligned with notation objects other than bar lines. These objects include `ambitus`, `breathing-sign`, `clef`, `custos`, `staff-bar`, `left-edge`, `key-cancellation`, `key-signature`, and `time-signature`.

Each type of object has its own default reference point, to which rehearsal marks are aligned:

```
% The rehearsal mark will be aligned to the right edge of the Clef
\override Score.RehearsalMark.break-align-symbols = #'(clef)
\key a \major
\clef treble
\mark "↓"
e'1
% The rehearsal mark will be aligned to the left edge of the Time Signature
\override Score.RehearsalMark.break-align-symbols = #'(time-signature)
\key a \major
\clef treble
\time 3/4
\mark "↓"
e'2.
% The rehearsal mark will be centered above the Breath Mark
\override Score.RehearsalMark.break-align-symbols = #'(breathing-sign)
```

```
\key a \major
\clef treble
\time 4/4
e'1
\breathes
\mark "↓"
```



A list of possible target alignment objects may be specified. If some of the objects are invisible at that point due to the setting of `break-visibility` or the explicit visibility settings for keys and clefs, the rehearsal mark or bar number is aligned to the first object in the list which is visible. If no objects in the list are visible the object is aligned to the bar line. If the bar line is invisible the object is aligned to the place where the bar line would be.

```
% The rehearsal mark will be aligned to the right edge of the Key Signature
\override Score.RehearsalMark.break-align-symbols = #'(key-signature clef)
\key a \major
\clef treble
\mark "↓"
e'1
% The rehearsal mark will be aligned to the right edge of the Clef
\set Staff.explicitKeySignatureVisibility = #all-invisible
\override Score.RehearsalMark.break-align-symbols = #'(key-signature clef)
\key a \major
\clef bass
\mark "↓"
gis,1
% The rehearsal mark will be centered above the Bar Line
\set Staff.explicitKeySignatureVisibility = #all-invisible
\set Staff.explicitClefVisibility = #all-invisible
\override Score.RehearsalMark.break-align-symbols = #'(key-signature clef)
\key a \major
\clef treble
\mark "↓"
e'1
```



The alignment of the rehearsal mark relative to the notation object can be changed, as shown in the following example. In a score with multiple staves, this setting should be done for all the staves.

```
% The RehearsalMark will be aligned with the right edge of the Key Signature
\override Score.RehearsalMark.break-align-symbols = #'(key-signature)
\key a \major
\clef treble
\time 4/4
```

```

\mark "↓"
e'1
% The RehearsalMark will be centered above the Key Signature
\once \override Score.KeySignature.break-align-anchor-alignment = #CENTER
\mark "↓"
\key a \major
e'1
% The RehearsalMark will be aligned with the left edge of the Key Signature
\once \override Score.KeySignature.break-align-anchor-alignment = #LEFT
\key a \major
\mark "↓"
e'1

```

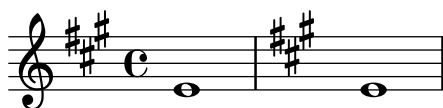


The rehearsal mark can also be offset to the right or left of the left edge by an arbitrary amount. The units are staff-spaces:

```

% The RehearsalMark will be aligned with the left edge of the Key Signature
% and then shifted right by 3.5 staff-spaces
\override Score.RehearsalMark.break-align-symbols = #'(key-signature)
\once \override Score.KeySignature.break-align-anchor = #3.5
\key a \major
\mark "↓"
e'1
% The RehearsalMark will be aligned with the left edge of the Key Signature
% and then shifted left by 2 staff-spaces
\once \override Score.KeySignature.break-align-anchor = #-2
\key a \major
\mark "↓"
e'1

```



### 5.5.2 Vertical grouping of grobs

The `VerticalAlignment` and `VerticalAxisGroup` grobs work together. `VerticalAxisGroup` groups together different grobs like `Staff`, `Lyrics`, etc. `VerticalAlignment` then vertically aligns the different grobs grouped together by `VerticalAxisGroup`. There is usually only one `VerticalAlignment` per score but every `Staff`, `Lyrics`, etc. has its own `VerticalAxisGroup`.

### 5.5.3 Modifying stencils

All layout objects have a `stencil` property which is part of the `grob-interface`. By default, this property is usually set to a function specific to the object that is tailor-made to render the symbol which represents it in the output. For example, the standard setting for the `stencil` property of the `MultiMeasureRest` object is `ly:multi-measure-rest::print`.



The standard symbol for any object can be replaced by modifying the `stencil` property to reference a different, specially-written, procedure. This requires a high level of knowledge of the internal workings of LilyPond, but there is an easier way which can often produce adequate results.

This is to set the `stencil` property to the procedure which prints text – `ly:text-interface::print` – and to add a `text` property to the object which is set to contain the markup text which produces the required symbol. Due to the flexibility of markup, much can be achieved – see in particular [\[Graphic notation inside markup\]](#), pagina [\[undefined\]](#).

The following example demonstrates this by changing the note head symbol to a cross within a circle.

```
Xin0 = {
  \once \override NoteHead.stencil = #ly:text-interface::print
  \once \override NoteHead.text = \markup {
    \combine
      \halign #-0.7 \draw-circle #0.85 #0.2 ##f
      \musicglyph #"noteheads.s2cross"
  }
}
\relative {
  a' a \Xin0 a a
}
```



Any of the glyphs in the feta Font can be supplied to the `\musicglyph` markup command – see [\[The Feta font\]](#), pagina [\[undefined\]](#).

## Vedi anche

Guida alla notazione: [\[Graphic notation inside markup\]](#), pagina [\[undefined\]](#), [\[Formatting text\]](#), pagina [\[undefined\]](#), [\[Text markup commands\]](#), pagina [\[undefined\]](#), [\[The Feta font\]](#), pagina [\[undefined\]](#).

### 5.5.4 Modifying shapes

#### Modifying ties and slurs

Ties, Slurs, PhrasingSlurs, LaissezVibrerTies and RepeatTies are all drawn as third-order Bézier curves. If the shape of the tie or slur which is calculated automatically is not optimum, the shape may be modified manually in two ways:

- by specifying the displacements to be made to the control points of the automatically calculated Bézier curve, or
- by explicitly specifying the positions of the four control points required to define the wanted curve.

Both methods are explained below. The first method is more suitable if only slight adjustments to the curve are required; the second may be better for creating curves which are related to just a single note.

### *Cubic Bézier curves*

Third-order or cubic Bézier curves are defined by four control points. The first and fourth control points are precisely the starting and ending points of the curve. The intermediate two control points define the shape. Animations showing how the curve is drawn can be found on the web, but the following description may be helpful. The curve starts from the first control point heading directly towards the second, gradually bending over to head towards the third and continuing to bend over to head towards the fourth, arriving there travelling directly from the third control point. The curve is entirely contained in the quadrilateral defined by the four control points. Translations, rotations and scaling of the control points all result in exactly the same operations on the curve.

### *Specifying displacements from current control points*

In this example the automatic placement of the tie is not optimum, and `\tieDown` would not help.

```
<<
  { e'1~ 1 }
\\
\relative { r4 <g' c,> <g c,> <g c,> }
>>
```



Adjusting the control points of the tie with `\shape` allows the collisions to be avoided.

The syntax of `\shape` is

```
[-]\shape displacements item
```

This will reposition the control-points of *item* by the amounts given by *displacements*. The *displacements* argument is a list of number pairs or a list of such lists. Each element of a pair represents the displacement of one of the coordinates of a control-point. If *item* is a string, the result is `\once\override` for the specified grob type. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied.

In other words, the `\shape` function can act as either a `\once\override` command or a `\tweak` command depending on whether the *item* argument is a grob name, like “Slur”, or a music expression, like “(”. The *displacements* argument specifies the displacements of the four control points as a list of four pairs of (dx . dy) values in units of staff-spaces (or a list of such lists if the curve has more than one segment).

The leading hyphen is required if and only if the `\tweak` form is being used.

So, using the same example as above and the `\once\override` form of `\shape`, this will raise the tie by half a staff-space:

```
<<
  {
    \shape #'((0 . 0.5) (0 . 0.5) (0 . 0.5) (0 . 0.5)) Tie
    e'1~ 1
  }
\\
\relative { r4 <g' c,> <g c,> <g c,> }
>>
```



This positioning of the tie is better, but maybe it should be raised more in the center. The following example does this, this time using the alternative `\tweak` form:

```
<<
{
  e'1-\shape #'((0 . 0.5) (0 . 1) (0 . 1) (0 . 0.5)) ~ e'
}
\\
\relative { r4 <g' c,> <g c,> <g c,> }
>>
```



Changes to the horizontal positions of the control points may be made in the same way, and two different curves starting at the same musical moment may also be shaped:

```
\relative {
  c''8(\( a) a'4 e c\)
  \shape #'((0.7 . -0.4) (0.5 . -0.4) (0.3 . -0.3) (0 . -0.2)) Slur
  \shape #'((0 . 0) (0 . 0.5) (0 . 0.5) (0 . 0)) PhrasingSlur
  c8(\( a) a'4 e c\)
}
```



The `\shape` function can also displace the control points of curves which stretch across line breaks. Each piece of the broken curve can be given its own list of offsets. If changes to a particular segment are not needed, the empty list can serve as a placeholder. In this example the line break makes the single slur look like two:

```
\relative {
  c'4( f g c
  \break
  d,4 c' f, c)
}
```



Changing the shapes of the two halves of the slur makes it clearer that the slur continues over the line break:

`% ()` may be used as a shorthand for `((0 . 0) (0 . 0) (0 . 0) (0 . 0))`

```
% if any of the segments does not need to be changed
\relative c' {
  \shape #'(
    (( 0 . 0) (0 . 0) (0 . 0) (0 . 1))
    ((0.5 . 1.5) (1 . 0) (0 . 0) (0 . -1.5))
  ) Slur
  c4( f g c
  \break
  d,4 c' f, c)
}
```



If an S-shaped curve is required the control points must always be adjusted manually — LilyPond will never select such shapes automatically.

```
\relative c'' {
  c8( e b-> f d' a e-> g)
  \shape #'((0 . -1) (5.5 . -0.5) (-5.5 . -10.5) (0 . -5.5)) PhrasingSlur
  c8\ ( e b-> f d' a e-> g\ )
}
```



### *Specifying control points explicitly*

The coordinates of the Bézier control points are specified in units of staff-spaces. The X coordinate is relative to the reference point of the note to which the tie or slur is attached, and the Y coordinate is relative to the staff center line. The coordinates are specified as a list of four pairs of decimal numbers (reals). One approach is to estimate the coordinates of the two end points, and then guess the two intermediate points. The optimum values are then found by trial and error. Be aware that these values may need to be manually adjusted if any further changes are made to the music or the layout.

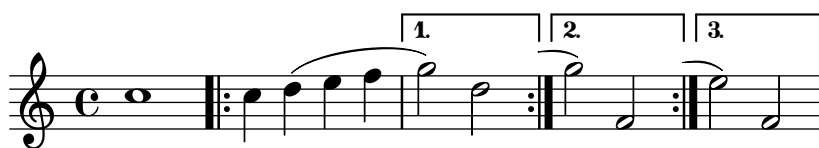
One situation where specifying the control points explicitly is preferable to specifying displacements is when they need to be specified relative to a single note. Here is an example of this. It shows one way of indicating a slur extending into alternative sections of a volta repeat.

```
\relative {
  c''1
  \repeat volta 3 { c4 d( e f )
  \alternative {
    { g2) d }
    {
      g2
      % create a slur and move it to a new position
    }
  }
}
```

```

% the <> is just an empty chord to carry the slur termination
-\tweak control-points #'((-2 . 3.8) (-1 . 3.9) (0 . 4) (1 . 3.4)) ( <> )
f,
}
{
e'2
% create a slur and move it to a new position
-\tweak control-points #'((-2 . 3) (-1 . 3.1) (0 . 3.2) (1 . 2.4)) ( <> )
f,
}
}
}

```



## Problemi noti e avvertimenti

It is not possible to modify shapes of ties or slurs by changing the `control-points` property if there are multiple ties or slurs at the same musical moment – the `\tweak` command will also not work in this case. However, the `tie-configuration` property of `TieColumn` can be overridden to set start line and direction as required.

## Vedi anche

Guida al funzionamento interno: Sezione “TieColumn” in *Guida al Funzionamento Interno*.

### 5.5.5 Modifying broken spanners

#### Using `\alterBroken`

When a spanner crosses a line break or breaks, each piece inherits the attributes of the original spanner. Thus, ordinary tweaking of a broken spanner applies the same modifications to each of its segments. In the example below, overriding `thickness` affects the slur on either side of the line break.

```

\relative c'' {
  r2
  \once\override Slur.thickness = 10
  c8( d e f
  \break
  g8 f e d) r2
}

```



Independently modifying the appearance of individual pieces of a broken spanner is possible with the `\alterBroken` command. This command can produce either an `\override` or a `\tweak` of a spanner property.

The syntax for `\alterBroken` is

```
[-]\alterBroken property values item
```

The argument *values* is a list of values, one for each broken piece. If *item* is a grob name like `Slur` or `Staff.PianoPedalBracket`, the result is an `\override` of the specified grob type. If *item* is a music expression such as “(” or “[” the result is the same music expression with an appropriate tweak applied.

The leading hyphen must be used with the `\tweak` form. Do not add it when `\alterBroken` is used as an `\override`.

In its `\override` usage, `\alterBroken` may be prefaced by `\once` or `\temporary` and reverted by using `\revert` with *property*.

The following code applies an independent `\override` to each of the slur segments in the previous example:

```
\relative c' {
  r2
  \alterBroken thickness #'(10 1) Slur
  c8( d e f
  \break
  g8 f e d) r2
}
```



The `\alterBroken` command may be used with any spanner object, including `Tie`, `PhrasingSlur`, `Beam` and `TextSpanner`. For example, an editor preparing a scholarly edition may wish to indicate the absence of part of a phrasing slur in a source by dashing only the segment which has been added. The following example illustrates how this can be done, in this case using the `\tweak` form of the command:

```
% The empty list is conveniently used below, because it is the
% default setting of dash-definition, resulting in a solid curve.
\relative {
  c''2-\alterBroken dash-definition #'(( ) ((0 1.0 0.4 0.75))) \(\e
  \break
  g2 e\ )
}
```



It is important to understand that `\alterBroken` will set each piece of a broken spanner to the corresponding value in *values*. When there are fewer values than pieces, any additional piece will be assigned the empty list. This may lead to undesired results if the layout property is not set to the empty list by default. In such cases, each segment should be assigned an appropriate value.

## Problemi noti e avvertimenti

Line breaks may occur in different places following changes in layout. Settings chosen for `\alterBroken` may be unsuitable for a spanner that is no longer broken or is split into more segments than before. Explicit use of `\break` can guard against this situation.

The `\alterBroken` command is ineffective for spanner properties accessed before line-breaking such as `direction`.

## Vedi anche

Estendere LilyPond: Sezione “Difficult tweaks” in *Estendere*.

### 5.5.6 Unpure-pure containers

Unpure-pure containers are useful for overriding *Y-axis* spacing calculations - specifically `Y-offset` and `Y-extent` - with a Scheme function instead of a literal (i.e. a number or pair).

For certain grobs, the `Y-extent` is based on the `stencil` property, overriding the `stencil` property of one of these will require an additional `Y-extent` override with an unpure-pure container. When a function overrides a `Y-offset` and/or `Y-extent` it is assumed that this will trigger line breaking calculations too early during compilation. So the function is not evaluated at all (usually returning a value of ‘0’ or ‘(0 . 0)’) which can result in collisions. A ‘pure’ function will not affect properties, objects or grob suicides and therefore will always have its Y-axis-related evaluated correctly.

Currently, there are about thirty functions that are already considered ‘pure’ and Unpure-pure containers are a way to set functions not on this list as ‘pure’. The ‘pure’ function is evaluated *before* any line-breaking and so the horizontal spacing can be adjusted ‘in time’. The ‘unpure’ function is then evaluated *after* line breaking.

**Nota:** As it is difficult to always know which functions are on this list we recommend that any ‘pure’ functions you create do not use `Beam` or `VerticalAlignment` grobs.

An unpure-pure container is constructed as follows;

```
(ly:make-unpure-pure-container f0 f1)
```

where `f0` is a function taking  $n$  arguments ( $n \geq 1$ ) and the first argument must always be the grob. This is the function that gives the actual result. `f1` is the function being labeled as ‘pure’ that takes  $n + 2$  arguments. Again, the first argument must always still be the grob but the second and third are ‘start’ and ‘end’ arguments.

*start* and *end* are, for all intents and purposes, dummy values that only matter for **Spanners** (i.e `Hairpin` or `Beam`), that can return different height estimations based on a starting and ending column.

The rest are the other arguments to the first function (which may be none if  $n = 1$ ).

The results of the second function are used as an approximation of the value needed which is then used by the first function to get the real value which is then used for fine-tuning much later during the spacing process.

```
#(define (square-line-circle-space grob)
  (let* ((pitch (ly:event-property (ly:grob-property grob 'cause) 'pitch))
```

```

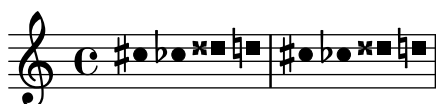
      (notename (ly:pitch-notename pitch)))
    (if (= 0 (modulo notename 2))
      (make-circle-stencil 0.5 0.0 #t)
      (make-filled-box-stencil '(0 . 1.0)
                               '(-0.5 . 0.5))))))

squareLineCircleSpace = {
  \override NoteHead.stencil = #square-line-circle-space
}

smartSquareLineCircleSpace = {
  \squareLineCircleSpace
  \override NoteHead.Y-extent =
    #(ly:make-unpure-pure-container
      ly:grob::stencil-height
      (lambda (grob start end) (ly:grob::stencil-height grob)))
}

\new Voice \with { \remove "Stem_engraver" }
\relative c'' {
  \squareLineCircleSpace
  cis4 ces disis d
  \smartSquareLineCircleSpace
  cis4 ces disis d
}

```



In the first measure, without the unpure-pure container, the spacing engine does not know the width of the note head and lets it collide with the accidentals. In the second measure, with unpure-pure containers, the spacing engine knows the width of the note heads and avoids the collision by lengthening the line accordingly.

Usually for simple calculations nearly-identical functions for both the ‘unpure’ and ‘pure’ parts can be used, by only changing the number of arguments passed to, and the scope of, the function. This use case is frequent enough that `ly:make-unpure-pure-container` constructs such a second function by default when called with only one function argument.

**Nota:** If a function is labeled as ‘pure’ and it turns out not to be, the results can be unexpected.

## 5.6 Uso delle funzioni musicali

Where tweaks need to be reused with different music expressions, it is often convenient to make the tweak part of a *music function*. In this section, we discuss only *substitution* functions, where the object is to substitute a variable into a piece of LilyPond input code. Other more complex functions are described in Sezione “Music functions” in *Estendere*.

### 5.6.1 Substitution function syntax

Making a function that substitutes a variable into LilyPond code is easy. The general form of these functions is

```
function =
```



```
#(define-music-function
  (arg1 arg2 ...)
  (type1? type2? ...)
  #{
    ...music...
  #})
```

where

*argN*                                      *nth* argument

*typeN?*                                      a scheme *type predicate* for which *argN* must return *#t*.

*...music...*                                      normal LilyPond input, using \$ (in places where only Lily-pond constructs are allowed) or # (to use it as a Scheme value or music function argument or music inside of music lists) to reference arguments (eg. '#arg1').

The list of type predicates is required. Some of the most common type predicates used in music functions are:

```
boolean?
cheap-list? (use instead of 'list?' for faster processing)
ly:duration?
ly:music?
ly:pitch?
markup?
number?
pair?
string?
symbol?
```

For a list of available type predicates, see [\(undefined\)](#) [Predefined type predicates], pagina [\(undefined\)](#). User-defined type predicates are also allowed.

## Vedi anche

Guida alla notazione: [\(undefined\)](#) [Predefined type predicates], pagina [\(undefined\)](#).

Estendere LilyPond: Sezione “Music functions” in *Estendere*.

File installati: `lily/music-scheme.cc`, `scm/c++.scm`, `scm/lily.scm`.

### 5.6.2 Substitution function examples

This section introduces some substitution function examples. These are not intended to be exhaustive, but rather to demonstrate some of the possibilities of simple substitution functions.

In the first example, a function is defined that simplifies setting the padding of a TextScript:

```
padText =
#(define-music-function
  (padding)
  (number?)
  #{
    \once \override TextScript.padding = #padding
  #})

\relative {
  c' '4^"piu mosso" b a b
  \padText #1.8
```

```

c4~"piu mosso" b a b
\padText #2.6
c4~"piu mosso" b a b
}

```



In addition to numbers, we can use music expressions such as notes for arguments to music functions:

```

custosNote =
#(define-music-function
  (note)
  (ly:music?)
  #{
    \tweak NoteHead.stencil #ly:text-interface::print
    \tweak NoteHead.text
      \markup \musicglyph #"custodes.mensural.u0"
    \tweak Stem.stencil ##f
    #note
  })
\relative { c'4 d e f \custosNote g }

```



Both of those functions are simple single expressions where only the last element of a function call or override is missing. For those particular function definitions, there is a simpler alternative syntax, namely just writing out the constant part of the expression and replacing its final missing element with `\etc`:

```

padText =
  \once \override TextScript.padding = \etc

\relative {
  c'4~"piu mosso" b a b
  \padText #1.8
  c4~"piu mosso" b a b
  \padText #2.6
  c4~"piu mosso" b a b
}

```



```

custosNote =
  \tweak NoteHead.stencil #ly:text-interface::print

```

```
\tweak NoteHead.text
  \markup \musicglyph #"custodes.mensural.u0"
\tweak Stem.stencil ##f
\etc
```

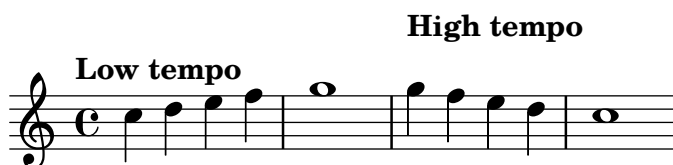
```
\relative { c'4 d e f \custosNote g }
```



Substitution functions with multiple arguments can be defined:

```
tempoPadded =
#(define-music-function
  (padding tempotext)
  (number? markup?)
  #{
    \once \override Score.MetronomeMark.padding = #padding
    \tempo \markup { \bold #tempotext }
  #})
```

```
\relative {
  \tempo \markup { "Low tempo" }
  c''4 d e f g1
  \tempoPadded #4.0 "High tempo"
  g4 f e d c1
}
```


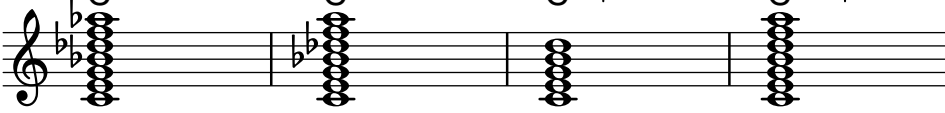
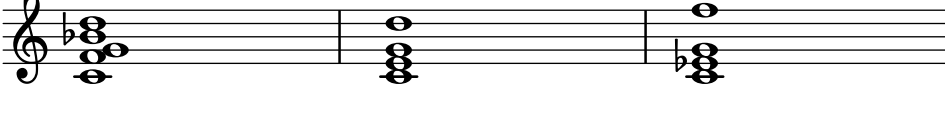



## Appendice A Tabelle del manuale della notazione

### A.1 Grafico dei nomi degli accordi



Il grafico seguente mostra due sistemi standard di rappresentazione dei nomi degli accordi, insieme alle altezze che li compongono.




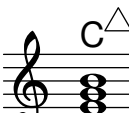




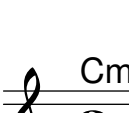
Ignatzek (default)	C	Cm	C+	C <sup>o</sup>
Alternative	C	C <sup>b3</sup>	C <sup>#5</sup>	C <sup>b3 b5</sup>
Def	C <sup>7</sup>	Cm <sup>7</sup>	C <sup>Δ</sup>	C <sup>o7</sup>
Alt	C <sup>7</sup>	C <sup>7 b3</sup>	C <sup>#7</sup>	C <sup>b3 b5 b7</sup>
Def	C <sup>7 #5</sup>	Cm <sup>Δ</sup>	C <sup>Δ #5</sup>	C <sup>∅</sup>
Alt	C <sup>7 #5</sup>	C <sup>b3 #7</sup>	C <sup>#5 #7</sup>	C <sup>7 b3 b5</sup>
Def	C <sup>6</sup>	Cm <sup>6</sup>	C <sup>9</sup>	Cm <sup>9</sup>
Alt	C <sup>6</sup>	C <sup>b3 6</sup>	C <sup>9</sup>	C <sup>9 b3</sup>
Def	Cm <sup>13</sup>	Cm <sup>11</sup>	Cm <sup>7 b5 9</sup>	C <sup>7 b9</sup>
Alt	C <sup>13 b3</sup>	C <sup>11 b3</sup>	C <sup>9 b3 b5</sup>	C <sup>7 b9</sup>
Def	C <sup>7 #9</sup>	C <sup>11</sup>	C <sup>7 #11</sup>	C <sup>13</sup>
Alt	C <sup>7 #9</sup>	C <sup>11</sup>	C <sup>9 #11</sup>	C <sup>13</sup>
Def	C <sup>7 #11 b13</sup>	C <sup>7 #5 #9</sup>	C <sup>7 #9 #11</sup>	C <sup>7 b13</sup>
Alt	C <sup>9 #11 b13</sup>	C <sup>7 #5 #9</sup>	C <sup>7 #9 #11</sup>	C <sup>11 b13</sup>

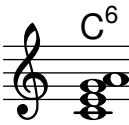

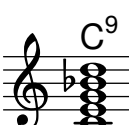
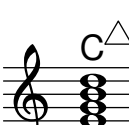


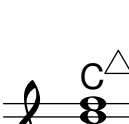
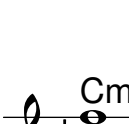
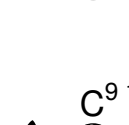
Def	$C^{7\flat 9\flat 13}$	$C^{7\sharp 11}$	$C^{\triangle 9}$	$C^{7\flat 13}$
Alt	$C^{11\flat 9\flat 13}$	$C^{9\sharp 11}$	$C^{9\sharp 7}$	$C^{11\flat 13}$
				
Def	$C^{7\flat 9\flat 13}$	$C^{7\flat 9\flat 13}$	$C^{\triangle 9}$	$C^{\triangle 13}$
Alt	$C^{11\flat 9\flat 13}$	$C^{13\flat 9}$	$C^{9\sharp 7}$	$C^{13\sharp 7}$
				
Def	$C^{\triangle \sharp 11}$	$C^{7\flat 9\flat 13}$	$C^{sus4}$	$C^{7sus4}$
Alt	$C^{9\sharp 7\sharp 11}$	$C^{13\flat 9}$	$C^{add4\ 5}$	$C^{add4\ 5\ 7}$
				
Def	$C^{9sus4}$	$C^9$	$C^{m11}$	
Alt	$C^{add4\ 5\ 7\ 9}$	$C^{add9}$	$C^{\flat 3\ add11}$	
				
Def	$C^{lyd}$	$C^{alt}$		
Alt	$C^{\sharp 7\ add\sharp 11}$	$C^{7\flat 9\flat 10\sharp 11\flat 13}$		
				

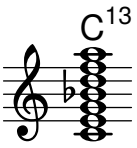
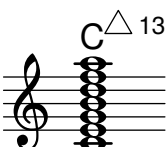
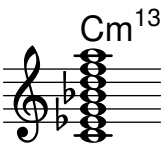




## A.2 Modificatori degli accordi

La tabella seguente mostra i modificatori degli accordi che possono essere usati per generare gli accordi comuni.

Tipo	Intervallo	Modificatore	Esempio	Output
Maggiore	Terza maggiore, quinta perfetta	5 o niente	c1:5	
Minore	Terza minore, quinta perfetta	m o m5	c1:m	

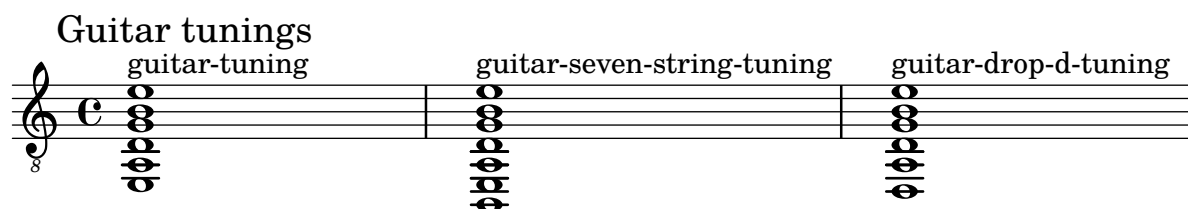
Aumentato	Terza maggiore, quinta aumentata	aug	c1:aug	
Diminuito	Terza minore, quinta diminuita	dim	c1:dim	
Settima dominante	Triade maggiore, settima minore	7	c1:7	
Settima maggiore	Triade maggiore, settima maggiore	maj7 o maj	c1:maj7	
Settima minore	Triade minore, settima minore	m7	c1:m7	
Settima diminuita	Triade diminuita, settima diminuita	dim7	c1:dim7	
Settima aumentata	Triade aumentata, settima minore	aug7	c1:aug	
Settima semidiminuita	Triade diminuita, settima minore	m7.5-	c1:m7.5-	
Minore di settima maggiore	Triade minore, settima maggiore	m7+	m7+	

Sesta maggiore	Triade maggiore, sesta	6	c1:6	
Sesta minore	Triade minore, sesta	m6	c1:m6	
Nona dominante	Settima dominante, nona maggiore	9	c1:9	
Nona maggiore	Settima maggiore, nona maggiore	maj9	c1:maj9	
Nona minore	Settima minore, nona maggiore	m9	c1:m9	
Undicesima dominante	Nona dominante, undicesima perfetta	11	c1:11	
Undicesima maggiore	Nona maggiore, undicesima perfetta	maj11	c1:maj11	
Undicesima minore	Nona minore, undicesima perfetta	m11	c1:m11	
Tredicesima dominante	Nona dominante, tredicesima maggiore	13	c1:13	

Tredicesima dominante	Undicesima dominante, tredicesima maggiore	13.11	c1:13.11	
Tredicesima maggiore	Undicesima maggiore, tredicesima maggiore	maj13.11	c1:maj13.11	
Tredicesima minore	Undicesima minore, tredicesima maggiore	m13.11	c1:m13.11	
Sospeso di seconda	Seconda maggiore, quinta perfetta	sus2	c1:sus2	
Sospeso di quarta	Quarta perfetta, quinta perfetta	sus4	c1:sus4	
Power chord (bicordo)	Quinta perfetta	1.5	\powerChords c1:5	
Power chord (tricordo)	Quinta perfetta, ottava	1.5.8	\powerChords c1:5.8	

### A.3 Accordature predefinite

Il grafico seguente mostra le accordature predefinite degli strumenti a corda.





4 guitar-drop-c-tuning guitar-open-g-tuning guitar-open-d-tuning

7 guitar-dadgad-tuning guitar-lute-tuning guitar-asus4-tuning

10 **Bass tunings**  
bass-tuning bass-four-string-tuning bass-drop-d-tuning

13 bass- ve-string-tuning bass-six-string-tuning

15 **Mandolin tunings**  
mandolin-tuning

16 **Banjo tunings**  
banjo-open-g-tuning banjo-c-tuning

18 banjo-modal-tuning banjo-open-d-tuning banjo-open-dm-tuning

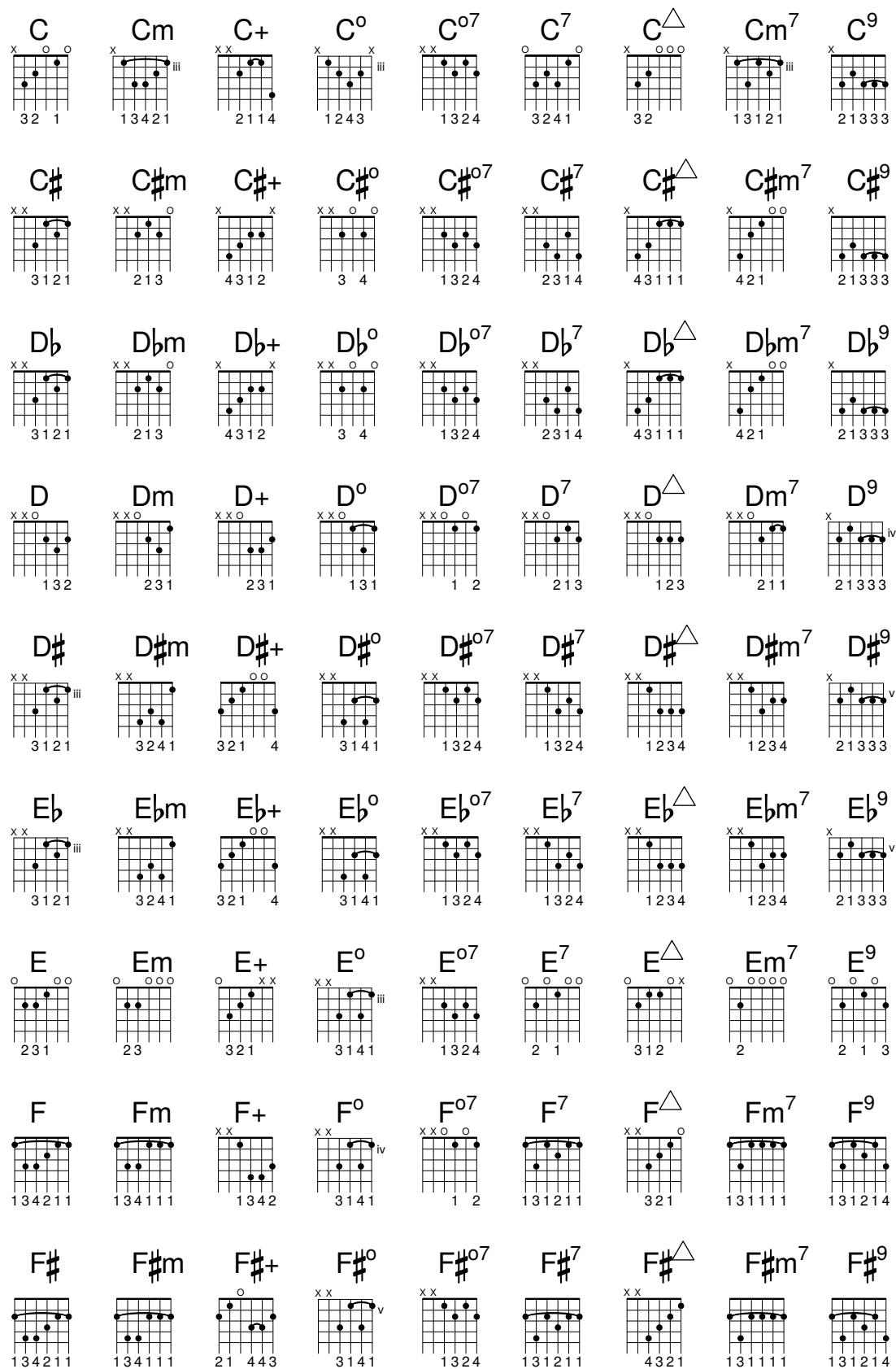
21 **Ukulele tunings**  
ukulele-tuning ukulele-d-tuning

23 tenor-ukulele-tuning baritone-ukulele-tuning

25 **Orchestral string tunings**  
violin-tuning viola-tuning cello-tuning double-bass-tuning

## A.4 Diagrammi degli accordi predefiniti

### Diagrammi per chitarra



 1 3 4 2 1 1	 1 3 4 1 1 1	 2 1 4 4 3	 3 1 4 1	 1 3 2 4	 1 3 1 2 1 1	 4 3 2 1	 1 3 1 1 1 1	 1 3 1 2 1 4
 2 1 3	 1 3 4 1 1 1	 1 3 4 2	 3 1 4 1	 1 3 2 4	 3 2 1	 4 3 2 1	 1 3 1 1 1 1	 1 3 1 2 1 4
 1 3 4 2 1 1	 1 3 4 1 1 1	 4 3 1 2	 3 1 4 1	 1 2	 1 3 1 2 1 1	 1 1 1 3	 1 3 1 1 1 1	 1 3 1 2 1 4
 1 3 4 2 1 1	 1 3 4 1 1 1	 4 3 1 2	 3 1 4 1	 1 2	 1 3 1 2 1 1	 1 1 1 3	 1 3 1 1 1 1	 1 3 1 2 1 4
 1 2 3	 2 3 1	 4 2 3 1	 1 2 3	 1 3 2 4	 1 3	 2 1 3	 2 1	 1 3 1 2 1 4
 1 2 3 4 1	 1 3 4 2 1	 2 1 4 4 3	 1 2 4 3	 1 3 2 4	 1 2 1 3 1	 1 3 2 4	 1 3 1 2 1	 1 3 1 2 1 4
 1 2 3 4 1	 1 3 4 2 1	 2 1 4 4 3	 1 2 4 3	 1 3 2 4	 1 2 1 3 1	 1 3 2 4	 1 3 1 2 1	 1 3 1 2 1 4
 1 2 3 4 1	 1 3 4 2 1	 2 1	 1 2 4 3	 1 2	 2 1 3 4	 1 3 2 4	 1 3 1 2 1	 2 1 3 3 3

## Diagrammi per ukulele

 3	 1 2 3	 1 4	 1 3 2 4	 1	 1 1 1 1	 1 2 2	 1 3	 2 1
-------	-----------	---------	-------------	-------	-------------	-----------	---------	---------

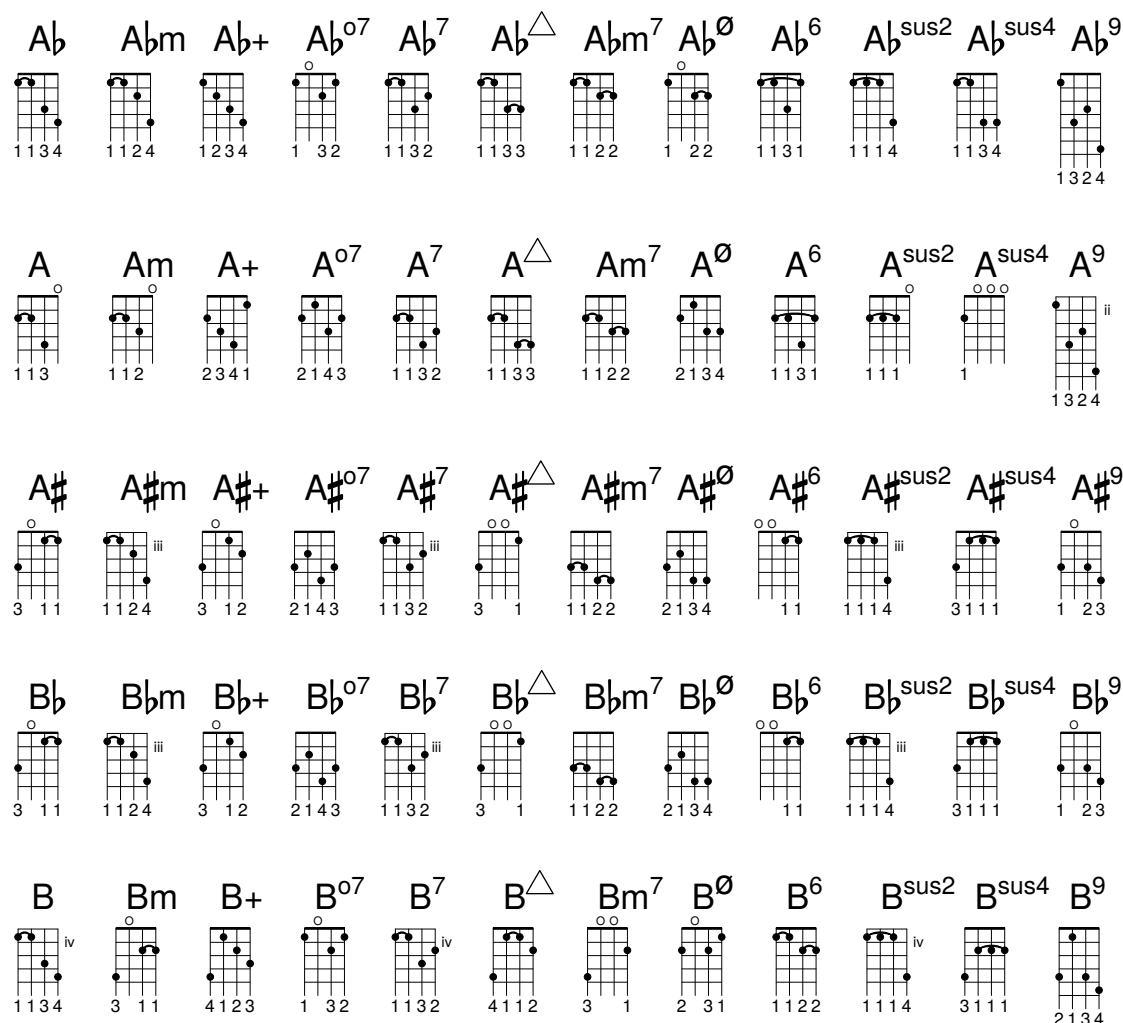
C <sup>#</sup>	C <sup>#</sup> m	C <sup>#</sup> +	C <sup>#</sup> <sup>o</sup>	C <sup>#</sup> <sup>7</sup>	C <sup>#</sup> <sup>△</sup>	C <sup>#</sup> m <sup>7</sup>	C <sup>#</sup> <sup>6</sup>	C <sup>#</sup> <sup>sus2</sup>	C <sup>#</sup> <sup>sus4</sup>	C <sup>#</sup> <sup>9</sup>
1 1 1 4	1 2 3 3	2 1 1 4	1 2	1 1 1 2	1 1 1 3	2 2 1 3	1 1 1 1	1 2 3 3	1 1 2 4	1 3 1 2
D <sup>b</sup>	D <sup>b</sup> m	D <sup>b</sup> +	D <sup>b</sup> <sup>o</sup>	D <sup>b</sup> <sup>7</sup>	D <sup>b</sup> <sup>△</sup>	D <sup>b</sup> m <sup>7</sup>	D <sup>b</sup> <sup>6</sup>	D <sup>b</sup> <sup>sus2</sup>	D <sup>b</sup> <sup>sus4</sup>	D <sup>b</sup> <sup>9</sup>
1 1 1 4	1 2 3 3	2 1 1 4	1 2	1 1 1 2	1 1 1 3	2 2 1 3	1 1 1 1	1 2 3 3	1 1 2 4	1 3 1 2
D	Dm	D+	D <sup>o</sup>	D <sup>7</sup>	D <sup>△</sup>	Dm <sup>7</sup>	D <sup>6</sup>	D <sup>sus2</sup>	D <sup>sus4</sup>	D <sup>9</sup>
1 2 3	2 2 1	2 1 1 4	1 3 2 4	1 1 1 2	1 1 1 3	2 2 1 3	1 1 1 1	1 2	1 2	1 3 1 2
D <sup>#</sup>	D <sup>#</sup> m	D <sup>#</sup> +	D <sup>#</sup> <sup>o</sup>	D <sup>#</sup> <sup>7</sup>	D <sup>#</sup> <sup>△</sup>	D <sup>#</sup> m <sup>7</sup>	D <sup>#</sup> <sup>6</sup>	D <sup>#</sup> <sup>sus2</sup>	D <sup>#</sup> <sup>sus4</sup>	D <sup>#</sup> <sup>9</sup>
2 2 1	3 3 2 1	2 2 1	1 3 1 4	1 1 1 2	1 2 1 2	2 2 1 4	1 1 1 1	2 2 1 1	2 3 4 1	1 1 1
E <sup>b</sup>	E <sup>b</sup> m	E <sup>b</sup> +	E <sup>b</sup> <sup>o</sup>	E <sup>b</sup> <sup>7</sup>	E <sup>b</sup> <sup>△</sup>	E <sup>b</sup> m <sup>7</sup>	E <sup>b</sup> <sup>6</sup>	E <sup>b</sup> <sup>sus2</sup>	E <sup>b</sup> <sup>sus4</sup>	E <sup>b</sup> <sup>9</sup>
2 2 1	3 3 2 1	2 2 1	1 3 1 4	1 1 1 2	1 2 1 2	2 2 1 4	1 1 1 1	2 2 1 1	2 3 4 1	1 1 1
E	Em	E+	E <sup>o</sup>	E <sup>7</sup>	E <sup>△</sup>	Em <sup>7</sup>	E <sup>6</sup>	E <sup>sus2</sup>	E <sup>sus4</sup>	E <sup>9</sup>
2 3 4 1	3 3 2 1	1 4	1 2	1 2 3	1 3 2	1 2	1 1 1 1	3 3 1 1	2 4 1	1 2 2 2
F	Fm	F+	F <sup>o</sup>	F <sup>7</sup>	F <sup>△</sup>	Fm <sup>7</sup>	F <sup>6</sup>	F <sup>sus2</sup>	F <sup>sus4</sup>	F <sup>9</sup>
2 1	1 2 4	2 1 1 4	1 3 2 4	2 3 1 4	2 4 1 3	1 3 2 4	2 2 1 4	1 3	3 1 1	1 2 2 2
F <sup>#</sup>	F <sup>#</sup> m	F <sup>#</sup> +	F <sup>#</sup> <sup>o</sup>	F <sup>#</sup> <sup>7</sup>	F <sup>#</sup> <sup>△</sup>	F <sup>#</sup> m <sup>7</sup>	F <sup>#</sup> <sup>6</sup>	F <sup>#</sup> <sup>sus2</sup>	F <sup>#</sup> <sup>sus4</sup>	F <sup>#</sup> <sup>9</sup>
3 1 2 1	2 1 3	2 1 1 4	1 3 2 4	3 4 2 1	2 4 1 3	1 3 2 4	2 2 1 4	1 1 2 4	4 1 2 3	1 2 2 2
G <sup>b</sup>	G <sup>b</sup> m	G <sup>b</sup> +	G <sup>b</sup> <sup>o</sup>	G <sup>b</sup> <sup>7</sup>	G <sup>b</sup> <sup>△</sup>	G <sup>b</sup> m <sup>7</sup>	G <sup>b</sup> <sup>6</sup>	G <sup>b</sup> <sup>sus2</sup>	G <sup>b</sup> <sup>sus4</sup>	G <sup>b</sup> <sup>9</sup>
3 1 2 1	2 1 3	2 1 1 4	1 3 2 4	3 4 2 1	2 4 1 3	1 3 2 4	2 2 1 4	1 1 2 4	4 1 2 3	1 2 2 2
G	Gm	G+	G <sup>o</sup>	G <sup>7</sup>	G <sup>△</sup>	Gm <sup>7</sup>	G <sup>6</sup>	G <sup>sus2</sup>	G <sup>sus4</sup>	G <sup>9</sup>
1 3 2	2 3 1	2 2 1	1 2	2 1 3	1 2 3	2 1 1	1 2	1 2	1 2 3	2 3 1 4

$G^\sharp$	$G^\sharp m$	$G^\sharp +$	$G^\sharp o$	$G^\sharp 7$	$G^\sharp \triangle$	$G^\sharp m^7$	$G^\sharp 6$	$G^\sharp \text{sus}2$	$G^\sharp \text{sus}4$	$G^\sharp 9$
3 1 2 1	1 3 4 2	1 4	1 3 2 4	1 3 2 4	1 2 3 3	1 4 2 3	1 3 2 4	2 3 4 1	1 3 3 3	1 3 2
$A^\flat$	$A^\flat m$	$A^\flat +$	$A^\flat o$	$A^\flat 7$	$A^\flat \triangle$	$A^\flat m^7$	$A^\flat 6$	$A^\flat \text{sus}2$	$A^\flat \text{sus}4$	$A^\flat 9$
3 1 2 1	1 3 4 2	1 4	1 3 2 4	1 3 2 4	1 2 3 3	1 4 2 3	1 3 2 4	2 3 4 1	1 3 3 3	1 3 2
$A$	$A m$	$A +$	$A o$	$A 7$	$A \triangle$	$A m^7$	$A 6$	$A \text{sus}2$	$A \text{sus}4$	$A 9$
2 1	1	2 1 1 4	1 3 2 4	1	1 2		1 3 2 4	2 3 4 1	1 2	1 2
$A^\sharp$	$A^\sharp m$	$A^\sharp +$	$A^\sharp o$	$A^\sharp 7$	$A^\sharp \triangle$	$A^\sharp m^7$	$A^\sharp 6$	$A^\sharp \text{sus}2$	$A^\sharp \text{sus}4$	$A^\sharp 9$
3 2 1 1	3 1 1 1	2 1 1 4	1 2	1 2 1 1	2 2 1 1	1 1 1 1	2 1 1	3 1 1	3 3 1 1	1 2 1 3
$B^\flat$	$B^\flat m$	$B^\flat +$	$B^\flat o$	$B^\flat 7$	$B^\flat \triangle$	$B^\flat m^7$	$B^\flat 6$	$B^\flat \text{sus}2$	$B^\flat \text{sus}4$	$B^\flat 9$
3 2 1 1	3 1 1 1	2 1 1 4	1 2	1 2 1 1	2 2 1 1	1 1 1 1	2 1 1	3 1 1	3 3 1 1	1 2 1 3
$B$	$B m$	$B +$	$B o$	$B 7$	$B \triangle$	$B m^7$	$B 6$	$B \text{sus}2$	$B \text{sus}4$	$B 9$
3 2 1 1	3 1 1 1	2 2 1	1 3 2 4	1 2 1 1	2 2 1 1	1 1 1 1	1 4 2 3	4 1 3 2	2 2 1 1	2 3 2 4

## Diagrammi per mandolino

$C$	$C m$	$C +$	$C^{o7}$	$C 7$	$C \triangle$	$C m^7$	$C^\emptyset$	$C 6$	$C \text{sus}2$	$C \text{sus}4$	$C 9$
4 1 2	1 1 2 4	4 1 2 3	2 1 4 3	4 2 1	4 1 1 2	1 1 2 2	3 1 4 2	1 1 2 2	3 1 1	3 1 1 1	1 3 2
$C^\sharp$	$C^\sharp m$	$C^\sharp +$	$C^\sharp o$	$C^\sharp 7$	$C^\sharp \triangle$	$C^\sharp m^7$	$C^\sharp^\emptyset$	$C^\sharp 6$	$C^\sharp \text{sus}2$	$C^\sharp \text{sus}4$	$C^\sharp 9$
4 2 3 1	2 3 1	4 1	2 1 1	4 2 1 3	4 1 1 2	1 1 2 2	3 1 4 2	1 1 2 2	1 1 3 4	3 1 1 1	2 1 3 4
$D^\flat$	$D^\flat m$	$D^\flat +$	$D^\flat o$	$D^\flat 7$	$D^\flat \triangle$	$D^\flat m^7$	$D^\flat^\emptyset$	$D^\flat 6$	$D^\flat \text{sus}2$	$D^\flat \text{sus}4$	$D^\flat 9$
4 2 3 1	2 3 1	4 1	2 1 1	4 2 1 3	4 1 1 2	1 1 2 2	3 1 4 2	1 1 2 2	1 1 3 4	3 1 1 1	2 1 3 4

D	Dm	D+	D <sup>o7</sup>	D <sup>7</sup>	D <sup>Δ</sup>	Dm <sup>7</sup>	D <sup>∅</sup>	D <sup>6</sup>	D <sup>sus2</sup>	D <sup>sus4</sup>	D <sup>9</sup>
1 2	2 1	3 12	1 32	1 32	1 42	2 31	1 32	1 23	1	1 2	421
D <sup>#</sup>	D <sup>#</sup> m	D <sup>#</sup> +	D <sup>#o7</sup>	D <sup>#7</sup>	D <sup>#Δ</sup>	D <sup>#m7</sup>	D <sup>#∅</sup>	D <sup>#6</sup>	D <sup>#sus2</sup>	D <sup>#sus4</sup>	D <sup>#9</sup>
3114	3112	123	2143	2143	2143	3142	2143	2134	3111	3114	2134
E <sup>b</sup>	E <sup>b</sup> m	E <sup>b</sup> +	E <sup>b o7</sup>	E <sup>b7</sup>	E <sup>bΔ</sup>	E <sup>b m7</sup>	E <sup>b ∅</sup>	E <sup>b6</sup>	E <sup>b sus2</sup>	E <sup>b sus4</sup>	E <sup>b9</sup>
3114	3112	123	2143	2143	2143	3142	2143	2134	3111	3114	2134
E	Em	E+	E <sup>o7</sup>	E <sup>7</sup>	E <sup>Δ</sup>	Em <sup>7</sup>	E <sup>∅</sup>	E <sup>6</sup>	E <sup>sus2</sup>	E <sup>sus4</sup>	E <sup>9</sup>
123	23	1234	2143	1 2	112	2	1	132	3111	31	2134
F	Fm	F+	F <sup>o7</sup>	F <sup>7</sup>	F <sup>Δ</sup>	Fm <sup>7</sup>	F <sup>∅</sup>	F <sup>6</sup>	F <sup>sus2</sup>	F <sup>sus4</sup>	F <sup>9</sup>
23 1	1341	1234	1 32	2131	2341	1131	1121	2 31	341	4211	2134
F <sup>#</sup>	F <sup>#</sup> m	F <sup>#</sup> +	F <sup># o7</sup>	F <sup>#7</sup>	F <sup>#Δ</sup>	F <sup># m7</sup>	F <sup># ∅</sup>	F <sup>#6</sup>	F <sup># sus2</sup>	F <sup># sus4</sup>	F <sup>#9</sup>
2341	1341	1234	2143	2131	2341	1131	1121	3142	3111	4211	213
G <sup>b</sup>	G <sup>b</sup> m	G <sup>b</sup> +	G <sup>b o7</sup>	G <sup>b7</sup>	G <sup>bΔ</sup>	G <sup>b m7</sup>	G <sup>b ∅</sup>	G <sup>b6</sup>	G <sup>b sus2</sup>	G <sup>b sus4</sup>	G <sup>b9</sup>
2341	1341	1234	2143	2131	2341	1131	1121	3142	3111	4211	213
G	Gm	G+	G <sup>o7</sup>	G <sup>7</sup>	G <sup>Δ</sup>	Gm <sup>7</sup>	G <sup>∅</sup>	G <sup>6</sup>	G <sup>sus2</sup>	G <sup>sus4</sup>	G <sup>9</sup>
12	13	123	2143	21	11	11	1121	2	3	11	1 4
G <sup>#</sup>	G <sup>#</sup> m	G <sup>#</sup> +	G <sup># o7</sup>	G <sup>#7</sup>	G <sup>#Δ</sup>	G <sup># m7</sup>	G <sup># ∅</sup>	G <sup>#6</sup>	G <sup># sus2</sup>	G <sup># sus4</sup>	G <sup>#9</sup>
1134	1124	1234	1 32	1132	1133	1122	1 22	1131	1114	1134	1324



## A.5 Formati carta predefiniti

I formati carta sono definiti in `scm/paper.scm`

### La serie A dell'“ISO 216”

"a10"	(26 x 37 mm)
"a9"	(37 x 52 mm)
"a8"	(52 x 74 mm)
"a7"	(74 x 105 mm)
"a6"	(105 x 148 mm)
"a5"	(148 x 210 mm)
"a4"	(210 x 297 mm)
"a3"	(297 x 420 mm)
"a2"	(420 x 594 mm)
"a1"	(594 x 841 mm)
"a0"	(841 x 1189 mm)

### La serie B dell'“ISO 216”

"b10"	(31 x 44 mm)
"b9"	(44 x 62 mm)

"b8"	(62 x 88 mm)
"b7"	(88 x 125 mm)
"b6"	(125 x 176 mm)
"b5"	(176 x 250 mm)
"b4"	(250 x 353 mm)
"b3"	(353 x 500 mm)
"b2"	(500 x 707 mm)
"b1"	(707 x 1000 mm)
"b0"	(1000 x 1414 mm)

**Due formati estesi, definiti nel “DIN 476”**

"4a0"	(1682 x 2378 mm)
"2a0"	(1189 x 1682 mm)

**La serie C standard dell’“ISO 269”**

"c10"	(28 x 40 mm)
"c9"	(40 x 57 mm)
"c8"	(57 x 81 mm)
"c7"	(81 x 114 mm)
"c6"	(114 x 162 mm)
"c5"	(162 x 229 mm)
"c4"	(229 x 324 mm)
"c3"	(324 x 458 mm)
"c2"	(458 x 648 mm)
"c1"	(648 x 917 mm)
"c0"	(917 x 1297 mm)

**Formati carta nordamericani**

"junior-legal"	(8.0 x 5.0 in)
"legal"	(8.5 x 14.0 in)
"ledger"	(17.0 x 11.0 in)
"letter"	(8.5 x 11.0 in)
"tabloid"	(11.0 x 17.0 in)
"11x17"	(11.0 x 17.0 in)
"17x11"	(17.0 x 11.0 in)

**Government-letter dell’IEEE Printer Working Group, per la scrittura dei bambini**

"government-letter"	(8 x 10.5 in)
---------------------	---------------



"government-legal"  
(8.5 x 13.0 in)

"philippine-legal"  
(8.5 x 13.0 in)

#### **Formati ANSI**

"ansi a" (8.5 x 11.0 in)

"ansi b" (17.0 x 11.0 in)

"ansi c" (17.0 x 22.0 in)

"ansi d" (22.0 x 34.0 in)

"ansi e" (34.0 x 44.0 in)

"engineering f"  
(28.0 x 40.0 in)

#### **Formati nordamericani per l'architettura**

"arch a" (9.0 x 12.0 in)

"arch b" (12.0 x 18.0 in)

"arch c" (18.0 x 24.0 in)

"arch d" (24.0 x 36.0 in)

"arch e" (36.0 x 48.0 in)

"arch e1" (30.0 x 42.0 in)

#### **Formati antichi ancora in uso nel Regno Unito**

"statement"  
(5.5 x 8.5 in)

"half letter"  
(5.5 x 8.5 in)

"quarto" (8.0 x 10.0 in)

"octavo" (6.75 x 10.5 in)

"executive"  
(7.25 x 10.5 in)

"monarch"  
(7.25 x 10.5 in)

"foolscap"  
(8.27 x 13.0 in)

"folio" (8.27 x 13.0 in)

"super-b"  
(13.0 x 19.0 in)

"post" (15.5 x 19.5 in)

"crown" (15.0 x 20.0 in)

"large post"  
(16.5 x 21.0 in)

"demy" (17.5 x 22.5 in)

"medium" (18.0 x 23.0 in)

"broadsheet"  
(18.0 x 24.0 in)

"royal" (20.0 x 25.0 in)

"elephant"  
(23.0 x 28.0 in)

"double demy"  
(22.5 x 35.0 in)

"quad demy"  
(35.0 x 45.0 in)

"atlas" (26.0 x 34.0 in)

"imperial"  
(22.0 x 30.0 in)

"antiquarian"  
(31.0 x 53.0 in)

#### **Formati basati su PA4**

"pa0" (840 x 1120 mm)

"pa1" (560 x 840 mm)

"pa2" (420 x 560 mm)

"pa3" (280 x 420 mm)

"pa4" (210 x 280 mm)

"pa5" (140 x 210 mm)

"pa6" (105 x 140 mm)

"pa7" (70 x 105 mm)

"pa8" (52 x 70 mm)

"pa9" (35 x 52 mm)

"pa10" (26 x 35 mm)

#### **Formato usato nel Sudest asiatico e in Australia**

"f4" (210 x 330 mm)

**Formato usato in esempi @lilypond molto piccoli della documentazione, basato sul formato a8 landscape.**

"a8landscape"  
(74 x 52 mm)

## A.6 Strumenti MIDI

Di seguito un elenco dei nomi che possono essere usati per la proprietà `midiInstrument`. L'ordine degli strumenti, a partire dalla colonna sinistra e proseguendo in basso, corrisponde ai 128 “numeri di programma” dello standard General MIDI.

acoustic grand	contrabass	lead 7 (fifths)
bright acoustic	tremolo strings	lead 8 (bass+lead)
electric grand	pizzicato strings	pad 1 (new age)
honky-tonk	orchestral harp	pad 2 (warm)
electric piano 1	timpani	pad 3 (polysynth)
electric piano 2	string ensemble 1	pad 4 (choir)
harpsichord	string ensemble 2	pad 5 (bowed)
clav	synthstrings 1	pad 6 (metallic)
celesta	synthstrings 2	pad 7 (halo)
glockenspiel	choir aahs	pad 8 (sweep)
music box	voice oohs	fx 1 (rain)
vibraphone	synth voice	fx 2 (soundtrack)
marimba	orchestra hit	fx 3 (crystal)
xylophone	trumpet	fx 4 (atmosphere)
tubular bells	trombone	fx 5 (brightness)
dulcimer	tuba	fx 6 (goblins)
drawbar organ	muted trumpet	fx 7 (echoes)
percussive organ	french horn	fx 8 (sci-fi)
rock organ	brass section	sitar
church organ	synthbrass 1	banjo
reed organ	synthbrass 2	shamisen
accordion	soprano sax	koto
harmonica	alto sax	kalimba
concertina	tenor sax	bagpipe
acoustic guitar (nylon)	baritone sax	fiddle
acoustic guitar (steel)	oboe	shanai
electric guitar (jazz)	english horn	tinkle bell
electric guitar (clean)	bassoon	agogo
electric guitar (muted)	clarinet	steel drums
overdriven guitar	piccolo	woodblock
distorted guitar	flute	taiko drum
guitar harmonics	recorder	melodic tom
acoustic bass	pan flute	synth drum
electric bass (finger)	blown bottle	reverse cymbal
electric bass (pick)	shakuhachi	guitar fret noise
fretless bass	whistle	breath noise
slap bass 1	ocarina	seashore
slap bass 2	lead 1 (square)	bird tweet
synth bass 1	lead 2 (sawtooth)	telephone ring
synth bass 2	lead 3 (calliope)	helicopter
violin	lead 4 (chiff)	applause
viola	lead 5 (charang)	gunshot
cello	lead 6 (voice)	

## A.7 Elenco dei colori

## Colori normali

La sintassi è descritta in [\[Coloring objects\]](#), pagina [\[Coloring objects\]](#).

black	white	red	green
blue	cyan	magenta	yellow
grey	darkred	darkgreen	darkblue
darkcyan	darkmagenta	darkyellow	

## Nomi di colore X

I nomi di colore X hanno diverse varianti:

Qualsiasi nome scritto come un'unica parola con lettere iniziali maiuscole (es: 'LightSlateBlue') può essere scritto anche con parole separate da spazio e senza maiuscola (es: 'light slate blue').

La parola 'grey' può essere sempre scritta come 'gray' (es: 'DarkSlateGray').

Alcuni nomi possono avere un suffisso numerico (es: 'LightSalmon4').

## Nomi di colori senza un suffisso numerico

snow	GhostWhite	WhiteSmoke	gainsboro	FloralWhite
OldLace	linen	AntiqueWhite	PapayaWhip	BlanchedAlmond
bisque	PeachPuff	NavajoWhite	moccasin	cornsilk
ivory	LemonChiffon	seashell	honeydew	MintCream
azure	AliceBlue	lavender	LavenderBlush	MistyRose
white	black	DarkSlateGrey	DimGrey	SlateGrey
LightSlateGrey	grey	LightGrey	MidnightBlue	navy
NavyBlue	CornflowerBlue	DarkSlateBlue	SlateBlue	MediumSlateBlue
LightSlateBlue	MediumBlue	RoyalBlue	blue	DodgerBlue
DeepSkyBlue	SkyBlue	LightSkyBlue	SteelBlue	LightSteelBlue
LightBlue	PowderBlue	PaleTurquoise	DarkTurquoise	MediumTurquoise
turquoise	cyan	LightCyan	CadetBlue	MediumAquamarine
aquamarine	DarkGreen	DarkOliveGreen	DarkSeaGreen	SeaGreen
MediumSeaGreen	LightSeaGreen	PaleGreen	SpringGreen	LawnGreen
green	chartreuse	MediumSpringGreen	GreenYellow	LimeGreen
YellowGreen	ForestGreen	OliveDrab	DarkKhaki	khaki
PaleGoldenrod	LightGoldenrodYellow	LightYellow	yellow	gold
LightGoldenrod	goldenrod	DarkGoldenrod	RosyBrown	IndianRed
SaddleBrown	sienna	peru	burlywood	beige
wheat	SandyBrown	tan	chocolate	firebrick
brown	DarkSalmon	salmon	LightSalmon	orange
DarkOrange	coral	LightCoral	tomato	OrangeRed
red	HotPink	DeepPink	pink	LightPink
PaleVioletRed	maroon	MediumVioletRed	VioletRed	magenta
violet	plum	orchid	MediumOrchid	DarkOrchid
DarkViolet	BlueViolet	purple	MediumPurple	thistle
DarkGrey	DarkBlue	DarkCyan	DarkMagenta	DarkRed
LightGreen				

## Nomi di colori con un suffisso numerico

Nei seguenti nomi il suffisso N può essere un numero compreso tra 1 e 4:

snowN	seashellN	AntiqueWhiteN	bisqueN	PeachPuffN
NavajoWhiteN	LemonChiffonN	cornsilkN	ivoryN	honeydewN
LavenderBlushN	MistyRoseN	azureN	SlateBlueN	RoyalBlueN

blueN	DodgerBlueN	SteelBlueN	DeepSkyBlueN	SkyBlueN
LightSkyBlueN	LightSteelBlueN	LightBlueN	LightCyanN	PaleTurquoiseN
CadetBlueN	turquoiseN	cyanN	aquamarineN	DarkSeaGreenN
SeaGreenN	PaleGreenN	SpringGreenN	greenN	chartreuseN
OliveDrabN	DarkOliveGreenN	khakiN	LightGoldenrodN	LightYellowN
yellowN	goldN	goldenrodN	DarkGoldenrodN	RosyBrownN
IndianRedN	siennaN	burlywoodN	wheatN	tanN
chocolateN	firebrickN	brownN	salmonN	LightSalmonN
orangeN	DarkOrangeN	coralN	tomatoN	OrangeRedN
redN	DeepPinkN	HotPinkN	pinkN	LightPinkN
PaleVioletRedN	maroonN	VioletRedN	magentaN	orchidN
plumN	MediumOrchidN	DarkOrchidN	purpleN	MediumPurpleN
thistleN				

## Scala di grigi

Una scala di grigi si ottiene con:

`greyN`

dove N è un numero compreso tra 0 e 100.

## A.8 Il tipo di carattere Feta

I simboli seguenti sono disponibili nel tipo di carattere Emmentaler e si possono utilizzare all'interno di un comando `\markup` specificando il nome del glifo, come mostrato nelle seguenti tabelle, per esempio `g~\markup {\musicglyph #"scripts.segno" }` o `\markup {\musicglyph #"five"}`. Maggiori informazioni in [\[Formatting text\]](#), pagina [\(undefined\)](#).

### Glifi della chiave

`clefs.C`



`clefs.C_change`



`clefs.varC`



`clefs.varC_change`



`clefs.F`



`clefs.F_change`



`clefs.G`



`clefs.G_change`



`clefs.GG`



`clefs.GG_change`



`clefs.tenorG`



`clefs.tenorG_change`



<code>clefs.percussion</code>	<b>  </b>	<code>clefs.percussion_change</code>	<b>  </b>
<code>clefs.varpercussion</code>	<b>□</b>	<code>clefs .varpercussion_change</code>	<b>□</b>
<code>clefs.tab</code>	<b><math>\mathcal{T}</math> <math>\mathcal{A}</math> <math>\mathcal{B}</math></b>	<code>clefs.tab_change</code>	<b><math>\mathcal{T}</math> <math>\mathcal{A}</math> <math>\mathcal{B}</math></b>

## Glifi delle indicazioni di tempo



















<code>timesig.C44</code>	<b>C</b>	<code>timesig.C22</code>	<b>¢</b>
--------------------------	----------	--------------------------	----------

## Glifi dei numeri







<code>plus</code>	<b>+</b>	<code>comma</code>	<b>,</b>
<code>hyphen</code>	<b>-</b>	<code>period</code>	<b>.</b>
<code>zero</code>	<b>0</b>	<code>one</code>	<b>1</b>
<code>two</code>	<b>2</b>	<code>three</code>	<b>3</b>
<code>four</code>	<b>4</b>	<code>ve</code>	<b>5</b>
<code>six</code>	<b>6</b>	<code>seven</code>	<b>7</b>
<code>eight</code>	<b>8</b>	<code>nine</code>	<b>9</b>

## Glifi delle alterazioni



















<code>accidentals.sharp</code>	<b>#</b>	<code>accidentals .sharp.arrowup</code>	<b>↑ #</b>
<code>accidentals .sharp.arrowdown</code>	<b>↓ #</b>	<code>accidentals .sharp.arrowboth</code>	<b>↑ # ↓</b>
<code>accidentals.sharp .slashslash.stem</code>	<b>‡</b>	<code>accidentals.sharp .slashslashslash.stemstem</code>	<b>‡</b>

<code>accidentals.sharp</code> <code>.slashslashslash.stem</code>		<code>accidentals.sharp</code> <code>.slashslash.stemstemstem</code>	
<code>accidentals.doublsharp</code>		<code>accidentals.natural</code>	
<code>accidentals</code> <code>.natural.arrowup</code>		<code>accidentals</code> <code>.natural.arrowdown</code>	
<code>accidentals</code> <code>.natural.arrowboth</code>		<code>accidentals. at</code>	
<code>accidentals. at.arrowup</code>		<code>accidentals</code> <code>. at.arrowdown</code>	
<code>accidentals</code> <code>. at.arrowboth</code>		<code>accidentals. at.slash</code>	
<code>accidentals. at</code> <code>.slashslash</code>		<code>accidentals</code> <code>.mirrored at. at</code>	
<code>accidentals.mirrored at</code>		<code>accidentals</code> <code>.mirrored at.backslash</code>	
<code>accidentals. at at</code>		<code>accidentals</code> <code>. at at.slash</code>	
<code>accidentals.rightparen</code>	)	<code>accidentals.leftparen</code>	(











## Glifi delle teste di nota predefinite

<code>noteheads.uM2</code>		<code>noteheads.dM2</code>	
<code>noteheads.sM1</code>		<code>noteheads.s0</code>	
<code>noteheads.s1</code>		<code>noteheads.s2</code>	

## Glifi delle teste di nota speciali



























<code>noteheads.sM1double</code>		<code>noteheads.s0diamond</code>	
<code>noteheads.s1diamond</code>		<code>noteheads.s2diamond</code>	
<code>noteheads.s0triangle</code>		<code>noteheads.d1triangle</code>	
<code>noteheads.u1triangle</code>		<code>noteheads.u2triangle</code>	
<code>noteheads.d2triangle</code>		<code>noteheads.s0slash</code>	
<code>noteheads.s1slash</code>		<code>noteheads.s2slash</code>	
<code>noteheads.s0cross</code>		<code>noteheads.s1cross</code>	
<code>noteheads.s2cross</code>		<code>noteheads.s2xcircle</code>	
<code>noteheads.s0harmonic</code>		<code>noteheads.s2harmonic</code>	

## Glifi delle teste di nota a forma variabile






















<code>noteheads.s0do</code>		<code>noteheads.d1do</code>	
<code>noteheads.u1do</code>		<code>noteheads.d2do</code>	
<code>noteheads.u2do</code>		<code>noteheads.s0doThin</code>	
<code>noteheads.d1doThin</code>		<code>noteheads.u1doThin</code>	
<code>noteheads.d2doThin</code>		<code>noteheads.u2doThin</code>	



noteheads.s0re	◐	noteheads.ulre	◐
noteheads.d1re	◐	noteheads.u2re	◐
noteheads.d2re	◐	noteheads.s0reThin	◐
noteheads.ulreThin	◐	noteheads.d1reThin	◐
noteheads.u2reThin	◐	noteheads.d2reThin	◐
noteheads.s0mi	◊	noteheads.s1mi	◊
noteheads.s2mi	◊	noteheads.s0miMirror	◊
noteheads.s1miMirror	◊	noteheads.s2miMirror	◊
noteheads.s0miThin	◊	noteheads.s1miThin	◊
noteheads.s2miThin	◊	noteheads.u0fa	◑
noteheads.d0fa	◑	noteheads.u1fa	◑
noteheads.d1fa	◑	noteheads.u2fa	◑
noteheads.d2fa	◑	noteheads.u0faThin	◑
noteheads.d0faThin	◑	noteheads.u1faThin	◑
noteheads.d1faThin	◑	noteheads.u2faThin	◑









<code>noteheads.d2faThin</code>		<code>noteheads.s0sol</code>	
<code>noteheads.s1sol</code>		<code>noteheads.s2sol</code>	
<code>noteheads.s0la</code>		<code>noteheads.s1la</code>	
<code>noteheads.s2la</code>		<code>noteheads.s0laThin</code>	
<code>noteheads.s1laThin</code>		<code>noteheads.s2laThin</code>	
<code>noteheads.s0ti</code>		<code>noteheads.ulti</code>	
<code>noteheads.d1ti</code>		<code>noteheads.u2ti</code>	
<code>noteheads.d2ti</code>		<code>noteheads.s0tiThin</code>	
<code>noteheads.ultiThin</code>		<code>noteheads.d1tiThin</code>	
<code>noteheads.u2tiThin</code>		<code>noteheads.d2tiThin</code>	
<code>noteheads.u0doFunk</code>		<code>noteheads.d0doFunk</code>	
<code>noteheads.u1doFunk</code>		<code>noteheads.d1doFunk</code>	
<code>noteheads.u2doFunk</code>		<code>noteheads.d2doFunk</code>	
<code>noteheads.u0reFunk</code>		<code>noteheads.d0reFunk</code>	
<code>noteheads.u1reFunk</code>		<code>noteheads.d1reFunk</code>	

noteheads.u2reFunk	►	noteheads.d2reFunk	◄
noteheads.u0miFunk	◊	noteheads.d0miFunk	◊
noteheads.u1miFunk	◊	noteheads.d1miFunk	◊
noteheads.s2miFunk	◆	noteheads.u0faFunk	▼
noteheads.d0faFunk	▴	noteheads.u1faFunk	▼
noteheads.d1faFunk	▴	noteheads.u2faFunk	▼
noteheads.d2faFunk	▴	noteheads.s0solFunk	○
noteheads.s1solFunk	○	noteheads.s2solFunk	●
noteheads.s0laFunk	□	noteheads.s1laFunk	□
noteheads.s2laFunk	■	noteheads.u0tiFunk	▷
noteheads.d0tiFunk	◁	noteheads.ultiFunk	▷
noteheads.d1tiFunk	◁	noteheads.u2tiFunk	►
noteheads.d2tiFunk	◄	noteheads.s0doWalker	▵
noteheads.u1doWalker	▿	noteheads.d1doWalker	▵
noteheads.u2doWalker	▿	noteheads.d2doWalker	▴



<code>noteheads.s0reWalker</code>		<code>noteheads.u1reWalker</code>	
<code>noteheads.d1reWalker</code>		<code>noteheads.u2reWalker</code>	
<code>noteheads.d2reWalker</code>		<code>noteheads.s0miWalker</code>	
<code>noteheads.s1miWalker</code>		<code>noteheads.s2miWalker</code>	
<code>noteheads.s0faWalker</code>		<code>noteheads.u1faWalker</code>	
<code>noteheads.d1faWalker</code>		<code>noteheads.u2faWalker</code>	
<code>noteheads.d2faWalker</code>		<code>noteheads.s0laWalker</code>	
<code>noteheads.s1laWalker</code>		<code>noteheads.s2laWalker</code>	
<code>noteheads.s0tiWalker</code>		<code>noteheads.ultiWalker</code>	
<code>noteheads.d1tiWalker</code>		<code>noteheads.u2tiWalker</code>	
<code>noteheads.d2tiWalker</code>			

## Glifi delle pause

<code>rests.0</code>		<code>rests.1</code>	
<code>rests.0o</code>		<code>rests.1o</code>	
<code>rests.M3</code>		<code>rests.M2</code>	
<code>rests.M1</code>		<code>rests.M1o</code>	

<code>rests.2</code>		<code>rests.2classical</code>	
<code>rests.2z</code>		<code>rests.3</code>	
<code>rests.4</code>		<code>rests.5</code>	
<code>rests.6</code>		<code>rests.7</code>	

## Glifi delle code

<code>ags.u3</code>		<code>ags.u4</code>	
<code>ags.u5</code>		<code>ags.u6</code>	
<code>ags.u7</code>		<code>ags.d3</code>	
<code>ags.d4</code>		<code>ags.d5</code>	
<code>ags.d6</code>		<code>ags.d7</code>	
<code>ags.ugrace</code>		<code>ags.dgrace</code>	





















## Glifi dei punti


<code>dots.dot</code>	
-----------------------	---





## Glifi delle dinamiche

space		f	<i><b>f</b></i>
m	<i><b>m</b></i>	p	<i><b>p</b></i>
r	<i><b>r</b></i>	s	<i><b>s</b></i>
z	<i><b>z</b></i>		









## Glifi dei segni

scripts.ufermata		scripts.dfermata	
scripts.ushortfermata		scripts.dshortfermata	
scripts.ulongfermata		scripts.dlongfermata	
scripts.uverylongfermata		scripts.dverylongfermata	
scripts.thumb		scripts.sforzato	
scripts.espr		scripts.staccato	
scripts.ustaccatissimo		scripts.dstaccatissimo	
scripts.tenuto		scripts.uportato	
scripts.dportato		scripts.umarcato	
scripts.dmarcato		scripts.open	

<code>scripts.halfopen</code>	ø	<code>scripts.halfopenvertical</code>	ø
<code>scripts.stopped</code>	+	<code>scripts.upbow</code>	V
<code>scripts.downbow</code>	⌞	<code>scripts.reverseturn</code>	∞
<code>scripts.turn</code>	∞	<code>scripts.trill</code>	<i>tr</i>
<code>scripts.upedalheel</code>	U	<code>scripts.dpedalheel</code>	∩
<code>scripts.upedaltoe</code>	V	<code>scripts.dpedaltoe</code>	∧
<code>scripts. ageolet</code>	○	<code>scripts.segno</code>	♫
<code>scripts.varsegno</code>		<code>scripts.coda</code>	⦿
<code>scripts.varcoda</code>	⦿	<code>scripts.rcomma</code>	,
<code>scripts.lcomma</code>	(	<code>scripts.rvarcomma</code>	/
<code>scripts.lvarcomma</code>	/	<code>scripts.arpeggio</code>	↗
<code>scripts.trill_element</code>	~	<code>scripts.arpeggio .arrow.M1</code>	↘
<code>scripts.arpeggio.arrow.1</code>	↗	<code>scripts.trilelement</code>	◆
<code>scripts.prall</code>	⚡	<code>scripts.mordent</code>	⚡
<code>scripts.prallprall</code>	⚡	<code>scripts.prallmordent</code>	⚡

<code>scripts.upprall</code>		<code>scripts.upmordent</code>	
<code>scripts.pralldown</code>		<code>scripts.downprall</code>	
<code>scripts.downmordent</code>		<code>scripts.prallup</code>	
<code>scripts.lineprall</code>		<code>scripts.caesura.curved</code>	
<code>scripts.caesura.straight</code>		<code>scripts.tickmark</code>	
<code>scripts.snappizzicato</code>		<code>scripts.ictus</code>	
<code>scripts.uaccentus</code>		<code>scripts.daccentus</code>	
<code>scripts.usemicirculus</code>		<code>scripts.dsemicirculus</code>	
<code>scripts.circulus</code>		<code>scripts.augmentum</code>	
<code>scripts</code> <code>.usignumcongruentiae</code>		<code>scripts</code> <code>.dsignumcongruentiae</code>	

## Glifi delle teste a forma di freccia








<code>arrowheads.open.01</code>		<code>arrowheads.open.0M1</code>	
<code>arrowheads.open.11</code>		<code>arrowheads.open.1M1</code>	
<code>arrowheads.close.01</code>		<code>arrowheads.close.0M1</code>	
<code>arrowheads.close.11</code>		<code>arrowheads.close.1M1</code>	



Glifi delle estremità delle parentesi

<code>brackettips.up</code>		<code>brackettips.down</code>	
-----------------------------	---	-------------------------------	---

Glifi dei pedali

<code>pedal.*</code>		<code>pedal.M</code>	
<code>pedal..</code>		<code>pedal.P</code>	
<code>pedal.d</code>		<code>pedal.e</code>	
<code>pedal.Ped</code>			
































Glifi della fisarmonica

<code>accordion.discant</code>		<code>accordion.dot</code>	
<code>accordion.freebass</code>		<code>accordion.stdbass</code>	
<code>accordion.bayanbass</code>		<code>accordion.oldEE</code>	
<code>accordion.push</code>		<code>accordion.pull</code>	
















Glifi delle legature di valore

<code>ties.lyric.short</code>		<code>ties.lyric.default</code>	
-------------------------------	---	---------------------------------	---













## Glifi della notazione vaticana





<code>clefs.vaticana.do</code>		<code>clefs.vaticana.do_change</code>	
<code>clefs.vaticana.fa</code>		<code>clefs.vaticana.fa_change</code>	
<code>custodes.vaticana.u0</code>		<code>custodes.vaticana.u1</code>	
<code>custodes.vaticana.u2</code>		<code>custodes.vaticana.d0</code>	
<code>custodes.vaticana.d1</code>		<code>custodes.vaticana.d2</code>	
<code>accidentals.vaticanaM1</code>		<code>accidentals.vaticana0</code>	
<code>dots.dotvaticana</code>		<code>noteheads .svaticana.punctum</code>	
<code>noteheads.svaticana .punctum.cavum</code>		<code>noteheads.svaticana .linea.punctum</code>	
<code>noteheads.svaticana .linea.punctum.cavum</code>		<code>noteheads.svaticana .inclinatum</code>	
<code>noteheads.svaticana.lpes</code>		<code>noteheads .svaticana.vlpes</code>	
<code>noteheads.svaticana.upes</code>		<code>noteheads .svaticana.vupes</code>	
<code>noteheads .svaticana.plica</code>		<code>noteheads .svaticana.vplica</code>	
<code>noteheads .svaticana.epiphonus</code>		<code>noteheads.svaticana .vepiphonus</code>	
<code>noteheads.svaticana .reverse.plica</code>		<code>noteheads.svaticana .reverse.vplica</code>	
<code>noteheads.svaticana .inner.cephalicus</code>		<code>noteheads.svaticana .cephalicus</code>	
<code>noteheads .svaticana.quilisma</code>			

## Glifi della notazione medicaea



























<code>clefs.medicaea.do</code>		<code>clefs.medicaea.do_change</code>	
<code>clefs.medicaea.fa</code>		<code>clefs.medicaea.fa_change</code>	
<code>custodes.medicaea.u0</code>		<code>custodes.medicaea.u1</code>	
<code>custodes.medicaea.u2</code>		<code>custodes.medicaea.d0</code>	
<code>custodes.medicaea.d1</code>		<code>custodes.medicaea.d2</code>	
<code>accidentals.medicaeaM1</code>		<code>noteheads.smedicaea.inclinatum</code>	
<code>noteheads.smedicaea.punctum</code>		<code>noteheads.smedicaea.rvirga</code>	
<code>noteheads.smedicaea.virga</code>			

## Glifi Hufnagel

<code>clefs.hufnagel.do</code>		<code>clefs.hufnagel.do_change</code>	
<code>clefs.hufnagel.fa</code>		<code>clefs.hufnagel.fa_change</code>	
<code>clefs.hufnagel.do.fa</code>		<code>clefs.hufnagel.do.fa_change</code>	
<code>custodes.hufnagel.u0</code>		<code>custodes.hufnagel.u1</code>	
<code>custodes.hufnagel.u2</code>		<code>custodes.hufnagel.d0</code>	
<code>custodes.hufnagel.d1</code>		<code>custodes.hufnagel.d2</code>	















accidentals.hufnagelM1		noteheads .shufnagel.punctum	
noteheads .shufnagel.virga		noteheads.shufnagel.lpes	

## Glifi della notazione mensurale

rests.M3mensural		rests.M2mensural	
rests.M1mensural		rests.0mensural	
rests.1mensural		rests.2mensural	
rests.3mensural		rests.4mensural	
clefs.mensural.c		clefs.mensural.c_change	
clefs.blackmensural.c		clefs.blackmensural .c_change	
clefs.mensural.f		clefs.mensural.f_change	
clefs.mensural.g		clefs.mensural.g_change	
custodes.mensural.u0		custodes.mensural.u1	
custodes.mensural.u2		custodes.mensural.d0	
custodes.mensural.d1		custodes.mensural.d2	
accidentals.mensural1		accidentals.mensuralM1	
ags.mensuralu03		ags.mensuralu13	
















ags.mensuralu23	}	ags.mensurald03	{
ags.mensurald13	{	ags.mensurald23	{
ags.mensuralu04	}	ags.mensuralu14	}
ags.mensuralu24	}	ags.mensurald04	{
ags.mensurald14	{	ags.mensurald24	{
ags.mensuralu05	}	ags.mensuralu15	}
ags.mensuralu25	}	ags.mensurald05	{
ags.mensurald15	{	ags.mensurald25	{
ags.mensuralu06	}	ags.mensuralu16	}
ags.mensuralu26	}	ags.mensurald06	{
ags.mensurald16	{	ags.mensurald26	{
timesig.mensural44	C	timesig.mensural22	♢
timesig.mensural32	O	timesig.mensural64	⊙
timesig.mensural94	⊙	timesig.mensural34	⊕

timesig.mensural68	¢	timesig.mensural198	ϕ
timesig.mensural48	⊙	timesig.mensural68alt	⊙
timesig.mensural24	Ⓢ	noteheads.uM3mensural	⏏
noteheads.dM3mensural	⏏	noteheads.sM3ligmensural	⏏
noteheads.uM2mensural	⏏	noteheads.dM2mensural	⏏
noteheads.sM2ligmensural	⏏	noteheads.sM1mensural	⏏
noteheads.urM3mensural	⏏	noteheads.drM3mensural	⏏
noteheads .srM3ligmensural	⏏	noteheads.urM2mensural	⏏
noteheads.drM2mensural	⏏	noteheads .srM2ligmensural	⏏
noteheads.srM1mensural	⏏	noteheads .uM3semimensural	⏏
noteheads .dM3semimensural	⏏	noteheads .sM3semiligmensural	⏏
noteheads .uM2semimensural	⏏	noteheads .dM2semimensural	⏏
noteheads .sM2semiligmensural	⏏	noteheads .sM1semimensural	⏏
noteheads .urM3semimensural	⏏	noteheads .drM3semimensural	⏏
noteheads .srM3semiligmensural	⏏	noteheads .urM2semimensural	⏏











noteheads .drM2semimensural		noteheads .srM2semiligmensural	
noteheads .srM1semimensural		noteheads .uM3blackmensural	
noteheads .dM3blackmensural		noteheads .sM3blackligmensural	
noteheads .uM2blackmensural		noteheads .dM2blackmensural	
noteheads .sM2blackligmensural		noteheads .sM1blackmensural	
noteheads.s0mensural		noteheads.s1mensural	
noteheads.s2mensural		noteheads .s0blackmensural	

## Glifi della notazione neomensurale

rests.M3neomensural		rests.M2neomensural	
rests.M1neomensural		rests.0neomensural	
rests.1neomensural		rests.2neomensural	
rests.3neomensural		rests.4neomensural	
clefs.neomensural.c		clefs.neomensural .c_change	
timesig.neomensural44		timesig.neomensural22	
timesig.neomensural32		timesig.neomensural64	
timesig.neomensural94		timesig.neomensural34	

timesig.neomensural68		timesig.neomensural98	
timesig.neomensural48		timesig.neomensural68alt	
timesig.neomensural24		noteheads.uM3neomensural	
noteheads.dM3neomensural		noteheads.uM2neomensural	
noteheads.dM2neomensural		noteheads.sM1neomensural	
noteheads .urM3neomensural		noteheads .drM3neomensural	
noteheads .urM2neomensural		noteheads .drM2neomensural	
noteheads .srM1neomensural		noteheads.s0neomensural	
noteheads.s1neomensural		noteheads.s2neomensural	








## Glifi Petrucci

clefs.petrucci.c1		clefs.petrucci.c1_change	
clefs.petrucci.c2		clefs.petrucci.c2_change	
clefs.petrucci.c3		clefs.petrucci.c3_change	
clefs.petrucci.c4		clefs.petrucci.c4_change	
clefs.petrucci.c5		clefs.petrucci.c5_change	



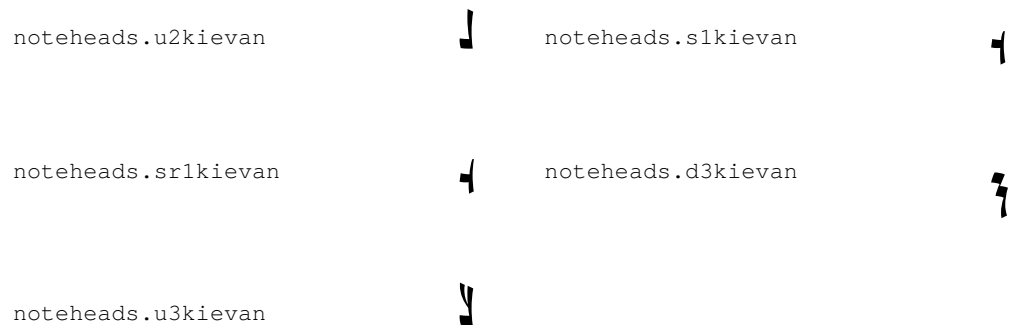
<code>clefs.petrucchi.f</code>		<code>clefs.petrucchi.f_change</code>	
<code>clefs.petrucchi.g</code>		<code>clefs.petrucchi.g_change</code>	
<code>noteheads.s0petrucci</code>		<code>noteheads.s1petrucci</code>	
<code>noteheads.s2petrucci</code>		<code>noteheads.s0blackpetrucci</code>	
<code>noteheads.s1blackpetrucci</code>		<code>noteheads.s2blackpetrucci</code>	

## Glifi Solesmes

<code>noteheads.ssolesmes.incl.parvum</code>		<code>noteheads.ssolesmes.auct.asc</code>	
<code>noteheads.ssolesmes.auct.desc</code>		<code>noteheads.ssolesmes.incl.auctum</code>	
<code>noteheads.ssolesmes.stropha</code>		<code>noteheads.ssolesmes.stropha.aucta</code>	
<code>noteheads.ssolesmes.oriscus</code>			

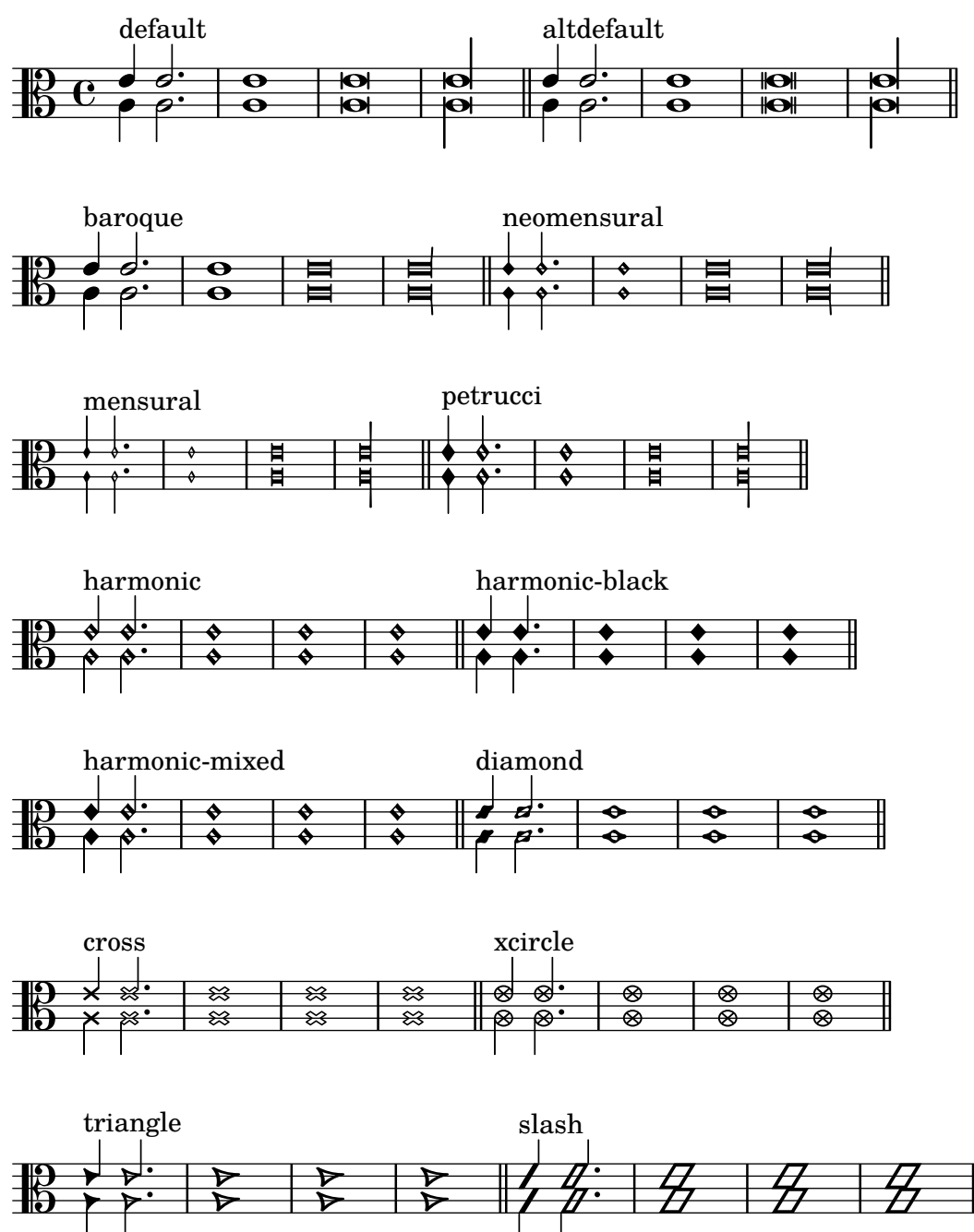
## Glifi della notazione di Kiev

<code>clefs.kievan.do</code>		<code>clefs.kievan.do_change</code>	
<code>accidentals.kievan1</code>		<code>accidentals.kievanM1</code>	
<code>scripts.barline.kievan</code>		<code>dots.dotkievan</code>	
<code>noteheads.sM2kievan</code>		<code>noteheads.sM1kievan</code>	
<code>noteheads.s0kievan</code>		<code>noteheads.d2kievan</code>	










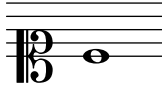

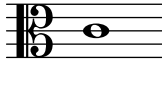
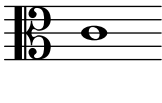
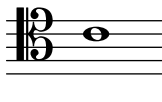
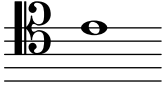

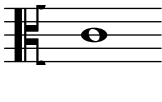



## A.9 Stili delle teste di nota

Si possono usare i seguenti stili per le teste di nota.



## A.10 Stili della chiave

La seguente tabella mostra tutti i diversi stili di chiave possibili (inclusi quelli in cui la posizione del *Do centrale*) cambia a seconda della chiave).

Esempio	Output	Esempio	Output
<code>\clef G</code>		<code>\clef "G2"</code>	
<code>\clef treble</code>		<code>\clef violin</code>	
<code>\clef french</code>		<code>\clef GG</code>	
<code>\clef tenorG</code>		<code>\clef soprano</code>	
<code>\clef mezzosoprano</code>		<code>\clef C</code>	
<code>\clef alto</code>		<code>\clef tenor</code>	
<code>\clef baritone</code>		<code>\clef varC</code>	
<code>\clef altovarC</code>		<code>\clef tenorvarC</code>	
<code>\clef baritonevarC</code>		<code>\clef varbaritone</code>	

`\clef baritonevarF`



`\clef F`



`\clef bass`



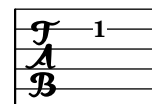
`\clef subbass`



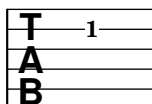
`\clef percussion`



`\new TabStaff {  
 \clef tab  
}`



`\new TabStaff {  
 \clef moderntab  
}`



## A.11 Comandi per *markup*

Tutti i comandi seguenti possono essere usati all'interno di `\markup`.

The following commands can all be used inside `\markup { }`.

### A.11.1 Font

`\abs-fontsize size (number) arg (markup)`

Use *size* as the absolute font size (in points) to display *arg*. Adjusts `baseline-skip` and `word-space` accordingly.

```
\markup {
  default text font size
  \hspace #2
  \abs-fontsize #16 { text font size 16 }
  \hspace #2
  \abs-fontsize #12 { text font size 12 }
}
```

default text font size    **text font size 16**    text font size 12

`\bold arg (markup)`

Switch to bold font-series.

```
\markup {
  default
  \hspace #2
  \bold
  bold
}
```

default    **bold**

`\box arg` (markup)

Draw a box round *arg*. Looks at `thickness`, `box-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \override #'(box-padding . 0.5)
  \box
  \line { V. S. }
}
```

V. S.

Used properties:

- `box-padding` (0.2)
- `font-size` (0)
- `thickness` (1)

`\caps arg` (markup)

Copy of the `\smallCaps` command.

```
\markup {
  default
  \hspace #2
  \caps {
    Text in small caps
  }
}
```

default TEXT IN SMALL CAPS

`\dynamic arg` (markup)

Use the dynamic font. This font only contains **s**, **f**, **m**, **z**, **p**, and **r**. When producing phrases, like ‘più **f**’, the normal words (like ‘più’) should be done in a different font. The recommended font for this is bold and italic.

```
\markup {
  \dynamic {
    sfzp
  }
}
```

***sfzp***

`\finger arg` (markup)

Set *arg* as small numbers.

```
\markup {
  \finger {
    1 2 3 4 5
  }
}
```

**1 2 3 4 5**

`\fontCaps arg` (markup)

Set `font-shape` to caps

Note: `\fontCaps` requires the installation and selection of fonts which support the caps font shape.

`\fontsize` *increment* (number) *arg* (markup)

Add *increment* to the font-size. Adjusts `baseline-skip` accordingly.

```
\markup {
  default
  \hspace #2
  \fontsize #-1.5
  smaller
}
```

`default`    `smaller`

Used properties:

- `baseline-skip` (2)
- `word-space` (1)
- `font-size` (0)

`\huge` *arg* (markup)

Set font size to +2.

```
\markup {
  default
  \hspace #2
  \huge
  huge
}
```

`default`    `huge`

`\italic` *arg* (markup)

Use italic `font-shape` for *arg*.

```
\markup {
  default
  \hspace #2
  \italic
  italic
}
```

`default`    `italic`

`\large` *arg* (markup)

Set font size to +1.

```
\markup {
  default
  \hspace #2
  \large
  large
}
```

`default`    `large`

`\larger arg` (markup)

Increase the font size relative to the current setting.

```
\markup {
  default
  \hspace #2
  \larger
  larger
}
```

**default larger**

`\magnify sz` (number) `arg` (markup)

Set the font magnification for its argument. In the following example, the middle A is 10% larger:

A `\magnify #1.1 { A }` A

Note: Magnification only works if a font name is explicitly selected. Use `\fontsize` otherwise.

```
\markup {
  default
  \hspace #2
  \magnify #1.5 {
    50% larger
  }
}
```

**default 50% larger**

`\medium arg` (markup)

Switch to medium font-series (in contrast to bold).

```
\markup {
  \bold {
    some bold text
    \hspace #2
    \medium {
      medium font series
    }
    \hspace #2
    bold again
  }
}
```

**some bold text** medium font series **bold again**

`\normal-size-sub arg` (markup)

Set `arg` in subscript with a normal font size.

```
\markup {
  default
  \normal-size-sub {
    subscript in standard size
  }
}
```

default subscript in standard size

Used properties:

- font-size (0)

`\normal-size-super arg` (markup)

Set *arg* in superscript with a normal font size.

```
\markup {
  default
  \normal-size-super {
    superscript in standard size
  }
}
```

default superscript in standard size

Used properties:

- font-size (0)

`\normal-text arg` (markup)

Set all font related properties (except the size) to get the default normal text font, no matter what font was used earlier.

```
\markup {
  \huge \bold \sans \caps {
    huge bold sans caps
    \hspace #2
    \normal-text {
      huge normal
    }
    \hspace #2
    as before
  }
}
```

**HUGE BOLD SANS CAPS** huge normal **AS BEFORE**

`\normalsize arg` (markup)

Set font size to default.

```
\markup {
  \teeny {
    this is very small
    \hspace #2
    \normalsize {
      normal size
    }
    \hspace #2
    teeny again
  }
}
```

this is very small    **normal size**    teeny again



`\number arg` (markup)

Set font family to `number`, which yields the font used for time signatures and fingerings. This font contains numbers and some punctuation; it has no letters.

```
\markup {
  \number {
    0 1 2 3 4 5 6 7 8 9 . ,
  }
}
```

**0123456789.,**

`\overtie arg` (markup)

Overtie *arg*.

```
\markup \line {
  \overtie "overtied"
  \override #'(offset . 5)
  \override #'(thickness . 1)
  \overtie "overtied"
  \override #'(offset . 1)
  \override #'(thickness . 5)
  \overtie "overtied"
}
```



Used properties:

- `shorten-pair` ((0 . 0))
- `direction` (1)
- `offset` (2)
- `thickness` (1)

`\replace replacements` (list) *arg* (markup)

Used to automatically replace a string by another in the markup *arg*. Each pair of the alist *replacements* specifies what should be replaced. The **key** is the string to be replaced by the **value** string.

```
\markup \replace #'(("thx" . "Thanks!")) thx
```

**Thanks!**

`\roman arg` (markup)

Set font family to `roman`.

```
\markup {
  \sans \bold {
    sans serif, bold
    \hspace #2
    \roman {
      text in roman font family
    }
    \hspace #2
    return to sans
  }
}
```

}

**sans serif, bold text in roman font family return to sans**

`\sans arg` (markup)

Switch to the sans serif font family.

```
\markup {
  default
  \hspace #2
  \sans {
    sans serif
  }
}
```

**default sans serif**

`\simple str` (string)

A simple text string; `\markup { foo }` is equivalent with `\markup { \simple #"foo" }`.

Note: for creating standard text markup or defining new markup commands, the use of `\simple` is unnecessary.

```
\markup {
  \simple #"simple"
  \simple #"text"
  \simple #"strings"
}
```

**simple text strings**

`\small arg` (markup)

Set font size to -1.

```
\markup {
  default
  \hspace #2
  \small
  small
}
```

**default small**

`\smallCaps arg` (markup)

Emit *arg* as small caps.

Note: `\smallCaps` does not support accented characters.

```
\markup {
  default
  \hspace #2
  \smallCaps {
    Text in small caps
  }
}
```

**default TEXT IN SMALL CAPS**

`\smaller arg` (markup)

Decrease the font size relative to the current setting.

```
\markup {
  \fontsize #3.5 {
    some large text
    \hspace #2
    \smaller {
      a bit smaller
    }
    \hspace #2
    more large text
  }
}
```

some large text a bit smaller more large text

`\sub arg` (markup)

Set *arg* in subscript.

```
\markup {
  \concat {
    H
    \sub {
      2
    }
    0
  }
}
```

H<sub>2</sub>O

Used properties:

- font-size (0)

`\super arg` (markup)

Set *arg* in superscript.

```
\markup {
  E =
  \concat {
    mc
    \super
    2
  }
}
```

E = mc<sup>2</sup>

Used properties:

- font-size (0)

`\teeny arg` (markup)

Set font size to -3.

```
\markup {
```

```

    default
    \hspace #2
    \teeny
    teeny
}

```

```

    default    teeny

```

**\text arg** (markup)

Use a text font instead of music symbol or music alphabet font.

```

\markup {
  \number {
    1, 2,
    \text {
      three, four,
    }
    5
  }
}

```

**1, 2**, three, four, **5**

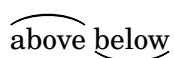
**\tie arg** (markup)

Adds a horizontal bow created with `make-tie-stencil` at bottom or top of *arg*. Looks at `thickness` to determine line thickness, and `offset` to determine y-offset. The added bow fits the extent of *arg*, `shorten-pair` may be used to modify this. *direction* may be set using an `override` or direction-modifiers or `voiceOne`, etc.

```

\markup {
  \override #'(direction . 1)
  \tie "above"
  \override #'(direction . -1)
  \tie "below"
}

```



Used properties:

- `shorten-pair` ((0 . 0))
- `direction` (1)
- `offset` (2)
- `thickness` (1)

**\tiny arg** (markup)

Set font size to -2.

```

\markup {
  default
  \hspace #2
  \tiny
  tiny
}

```

**default**    tiny

`\typewriter` *arg* (markup)

Use **font-family** typewriter for *arg*.

```
\markup {
  default
  \hspace #2
  \typewriter
  typewriter
}
```

**default**    typewriter

`\underline` *arg* (markup)

Underline *arg*. Looks at **thickness** to determine line thickness, and **offset** to determine line y-offset.

```
\markup \fill-line {
  \underline "underlined"
  \override #'(offset . 5)
  \override #'(thickness . 1)
  \underline "underlined"
  \override #'(offset . 1)
  \override #'(thickness . 5)
  \underline "underlined"
}
```

underlined

underlined

underlined

Used properties:

- **offset** (2)
- **thickness** (1)

`\undertie` *arg* (markup)

```
\markup \line {
  \undertie "undertied"
  \override #'(offset . 5)
  \override #'(thickness . 1)
  \undertie "undertied"
  \override #'(offset . 1)
  \override #'(thickness . 5)
  \undertie "undertied"
}
```

undertied undertied undertied

Used properties:

- **shorten-pair** ((0 . 0))
- **direction** (1)
- **offset** (2)
- **thickness** (1)

`\upright arg` (markup)

Set `font-shape` to `upright`. This is the opposite of `italic`.

```
\markup {
  \italic {
    italic text
    \hspace #2
    \upright {
      upright text
    }
    \hspace #2
    italic again
  }
}
```

*italic text*    **upright text**    *italic again*

### A.11.2 Align

`\center-align arg` (markup)

Align `arg` to its X center.

```
\markup {
  \column {
    one
    \center-align
    two
    three
  }
}
```

one  
two  
three

`\center-column args` (markup list)

Put `args` in a centered column.

```
\markup {
  \center-column {
    one
    two
    three
  }
}
```

one  
two  
three

Used properties:

- `baseline-skip`

`\column args` (markup list)

Stack the markups in `args` vertically. The property `baseline-skip` determines the space between markups in `args`.

```
\markup {
```

```

\column {
  one
  two
  three
}

```

```

one
two
three

```

Used properties:

- `baseline-skip`

`\combine` *arg1* (markup) *arg2* (markup)

Print two markups on top of each other.

Note: `\combine` cannot take a list of markups enclosed in curly braces as an argument; for this purpose use `\overlay` instead.

```

\markup {
  \fontsize #5
  \override #'(thickness . 2)
  \combine
    \draw-line #'(0 . 4)
    \arrow-head #Y #DOWN ##f
}

```



`\concat` *args* (markup list)

Concatenate *args* in a horizontal line, without spaces in between. Strings and simple markups are concatenated on the input level, allowing ligatures. For example, `\concat { "f" \simple #"i" }` is equivalent to `"fi"`.

```

\markup {
  \concat {
    one
    two
    three
  }
}

```

```

onetwothree

```

`\dir-column` *args* (markup list)

Make a column of *args*, going up or down, depending on the setting of the `direction` layout property.

```

\markup {
  \override #'(direction . ,UP) {
    \dir-column {
      going up
    }
  }
}

```

```

\hspace #1
\dir-column {
  going down
}
\hspace #1
\override #'(direction . 1) {
  \dir-column {
    going up
  }
}
}

```

```

up      up
going going going
      down

```

Used properties:

- `baseline-skip`
- `direction`

`\fill-line` *args* (markup list)

Put *markups* in a horizontal line of width *line-width*. The markups are spaced or flushed to fill the entire line. If there are no arguments, return an empty stencil.

```

\markup {
  \column {
    \fill-line {
      Words evenly spaced across the page
    }
  }
  \null
  \fill-line {
    \line { Text markups }
    \line {
      \italic { evenly spaced }
    }
    \line { across the page }
  }
}
}

```

```

Words      evenly      spaced      across      the      page

```

```

Text markups      evenly spaced      across the page

```

Used properties:

- `line-width` (#f)
- `word-space` (0.6)
- `text-direction` (1)

`\fill-with-pattern` *space* (number) *dir* (direction) *pattern* (markup) *left* (markup) *right* (markup)

Put *left* and *right* in a horizontal line of width *line-width* with a line of markups *pattern* in between. Patterns are spaced apart by *space*. Patterns are aligned to the *dir* markup.



```
\markup \column {
  "right-aligned :"
  \fill-with-pattern #1 #RIGHT . first right
  \fill-with-pattern #1 #RIGHT . second right
  \null
  "center-aligned :"
  \fill-with-pattern #1.5 #CENTER - left right
  \null
  "left-aligned :"
  \override #'(line-width . 50)
  \fill-with-pattern #2 #LEFT : left first
  \override #'(line-width . 50)
  \fill-with-pattern #2 #LEFT : left second
}
```

right-aligned :

rst	.....	right
second	.....	right

center-aligned :

left	- - - - -	right
------	-----------	-------

left-aligned :

left:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	rst
left:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	second

Used properties:

- line-width
- word-space

`\general-align` *axis* (integer) *dir* (number) *arg* (markup)

Align *arg* in *axis* direction to the *dir* side.

```
\markup {
  \column {
    one
    \general-align #X #LEFT
    two
    three
    \null
    one
    \general-align #X #CENTER
    two
    three
    \null
    \line {
      one
      \general-align #Y #UP
      two
      three
    }
  }
  \null
}
```

```

\line {
  one
  \general-align #Y #3.2
  two
  three
}
}
}

```

```

one
two
three

```

```

one
two
three

```

```

one two three

```

```

one three
two

```

`\halign` *dir* (number) *arg* (markup)

Set horizontal alignment. If *dir* is -1, then it is left-aligned, while +1 is right. Values in between interpolate alignment accordingly.

```

\markup {
  \column {
    one
    \halign #LEFT
    two
    three
    \null
    one
    \halign #CENTER
    two
    three
    \null
    one
    \halign #RIGHT
    two
    three
    \null
    one
    \halign #-5
    two
    three
  }
}

```

one  
two  
three

one  
two  
three

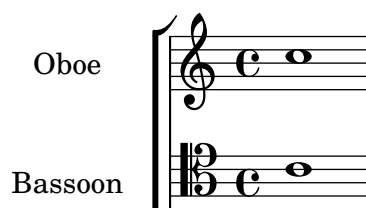
one  
two  
three

one  
two  
three

`\hcenter-in` *length* (number) *arg* (markup)

Center *arg* horizontally within a box of extending *length*/2 to the left and right.

```
\new StaffGroup <<
  \new Staff {
    \set Staff.instrumentName = \markup {
      \hcenter-in #12
      Oboe
    }
    c''1
  }
  \new Staff {
    \set Staff.instrumentName = \markup {
      \hcenter-in #12
      Bassoon
    }
    \clef tenor
    c'1
  }
>>
```



`\hspace` *amount* (number)

Create an invisible object taking up horizontal space *amount*.

```
\markup {
  one
  \hspace #2
  two
  \hspace #8
  three
}
```

one    two            three

`\justify-field` *symbol* (*symbol*)

Justify the data which has been assigned to *symbol*.

```
\header {
  title = "My title"
  myText = "Lorem ipsum dolor sit amet, consectetur adipisicing
    elit, sed do eiusmod tempor incididunt ut labore et dolore magna
    aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat."
}

\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \justify-field #'header:myText
    }
  }
}

\markup {
  \null
}
```

My title

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

`\justify-line` *args* (*markup list*)

Put *markups* in a horizontal line of width *line-width*. The markups are spread to fill the entire line and separated by equal space. If there are no arguments, return an empty stencil.

```
\markup {
  \justify-line {
    Space between neighboring words is constant
  }
}
```

Space            between            neighboring            words            is            constant

Used properties:

- `line-width` (`#f`)
- `word-space` (0.6)
- `text-direction` (1)

`\justify` *args* (markup list)

Like `\wordwrap`, but with lines stretched to justify the margins. Use `\override #'(line-width . X)` to set the line width; *X* is the number of staff spaces.

```
\markup {
  \justify {
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat.
  }
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (*#f*)
- `baseline-skip`

`\justify-string` *arg* (string)

Justify a string. Paragraphs may be separated with double newlines

```
\markup {
  \override #'(line-width . 40)
  \justify-string #"Lorem ipsum dolor sit amet, consectetur
    adipisicing elit, sed do eiusmod tempor incididunt ut labore
    et dolore magna aliqua.

    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat.

    Excepteur sint occaecat cupidatat non proident, sunt in culpa
    qui officia deserunt mollit anim id est laborum"
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Excepteur sint occaecat cupidatat non proident, sunt in culpa qui o cia deserunt mollit anim id est laborum

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`
- `baseline-skip`

`\left-align` *arg* (markup)

Align *arg* on its left edge.

```
\markup {
  \column {
    one
    \left-align
    two
    three
  }
}
```

```
one
two
three
```

`\left-column` *args* (markup list)

Put *args* in a left-aligned column.

```
\markup {
  \left-column {
    one
    two
    three
  }
}
```

```
one
two
three
```

Used properties:

- `baseline-skip`

`\line` *args* (markup list)

Put *args* in a horizontal line. The property `word-space` determines the space between markups in *args*.

```
\markup {
  \line {
    one two three
  }
}
```

```
one two three
```

Used properties:

- `text-direction` (1)
- `word-space`

`\lower` *amount* (number) *arg* (markup)

Lower *arg* by the distance *amount*. A negative *amount* indicates raising; see also `\raise`.

```
\markup {
  one
  \lower #3
  two
  three
}
```

one      three  
         two

`\overlay` *args* (markup list)

Takes a list of markups combining them.

```
\markup {
  \fontsize #5
  \override #'(thickness . 2)
  \overlay {
    \draw-line #'(0 . 4)
    \arrow-head #Y #DOWN ##f
    \translate #'(0 . 4)\arrow-head #Y #UP ##f
  }
}
```



`\pad-around` *amount* (number) *arg* (markup)

Add padding *amount* all around *arg*.

```
\markup {
  \box {
    default
  }
  \hspace #2
  \box {
    \pad-around #0.5 {
      padded
    }
  }
}
```

default	padded
---------	--------

`\pad-markup` *amount* (number) *arg* (markup)

Add space around a markup object. Identical to `\pad-around`.

```
\markup {
  \box {
    default
  }
  \hspace #2
```

```
\box {
  \pad-markup #1 {
    padded
  }
}
```

default	padded
---------	--------

`\pad-to-box` *x-ext* (pair of numbers) *y-ext* (pair of numbers) *arg* (markup)  
 Make *arg* take at least *x-ext*, *y-ext* space.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-to-box #'(0 . 10) #'(0 . 3) {
      padded
    }
  }
}
```

default	padded
---------	--------

`\pad-x` *amount* (number) *arg* (markup)  
 Add padding *amount* around *arg* in the X direction.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-x #2 {
      padded
    }
  }
}
```

default	padded
---------	--------

`\put-adjacent` *axis* (integer) *dir* (direction) *arg1* (markup) *arg2* (markup)  
 Put *arg2* next to *arg1*, without moving *arg1*.

`\raise` *amount* (number) *arg* (markup)

Raise *arg* by the distance *amount*. A negative *amount* indicates lowering, see also `\lower`.

The argument to `\raise` is the vertical displacement amount, measured in (global) staff spaces. `\raise` and `\super` raise objects in relation to their surrounding markups.



If the text object itself is positioned above or below the staff, then `\raise` cannot be used to move it, since the mechanism that positions it next to the staff cancels any shift made with `\raise`. For vertical positioning, use the `padding` and/or `extra-offset` properties.

```
\markup {
  C
  \small
  \bold
  \raise #1.0
  9/7+
}
```

**C 9/7+**

`\right-align arg` (markup)

Align *arg* on its right edge.

```
\markup {
  \column {
    one
    \right-align
    two
    three
  }
}
```

one  
two  
three

`\right-column args` (markup list)

Put *args* in a right-aligned column.

```
\markup {
  \right-column {
    one
    two
    three
  }
}
```

one  
two  
three

Used properties:

- `baseline-skip`

`\rotate ang` (number) *arg* (markup)

Rotate object with *ang* degrees around its center.

```
\markup {
  default
  \hspace #2
  \rotate #45
}
```

```
\line {
  rotated 45°
}
```

default

rotated 45°

`\translate` *offset* (pair of numbers) *arg* (markup)

Translate *arg* relative to its surroundings. *offset* is a pair of numbers representing the displacement in the X and Y axis.

```
\markup {
  *
  \translate #'(2 . 3)
  \line { translated two spaces right, three up }
}
```

translated two spaces right, three up

\*

`\translate-scaled` *offset* (pair of numbers) *arg* (markup)

Translate *arg* by *offset*, scaling the offset by the `font-size`.

```
\markup {
  \fontsize #5 {
    * \translate #'(2 . 3) translate
    \hspace #2
    * \translate-scaled #'(2 . 3) translate-scaled
  }
}
```

\*

translate

\*

translate-scaled

Used properties:

- `font-size` (0)

`\vcenter` *arg* (markup)

Align *arg* to its Y center.

```
\markup {
  one
  \vcenter
  two
  three
}
```

one two three

`\vspace` *amount* (number)

Create an invisible object taking up vertical space of *amount* multiplied by 3.

```
\markup {
```

```

\center-column {
one
\vspace #2
two
\vspace #5
three
}
}
one

```

two

three

`\wordwrap-field` *symbol* (symbol)

Wordwrap the data which has been assigned to *symbol*.

```

\header {
title = "My title"
myText = "Lorem ipsum dolor sit amet, consectetur adipisicing
elit, sed do eiusmod tempor incididunt ut labore et dolore
magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat."
}

```

```

\paper {
bookTitleMarkup = \markup {
\column {
\fill-line { \fromproperty #'header:title }
\null
\wordwrap-field #'header:myText
}
}
}

```

```

\markup {
\null
}

```

My title

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

`\wordwrap` *args* (markup list)

Simple wordwrap. Use `\override #'(line-width . X)` to set the line width, where *X* is the number of staff spaces.

```
\markup {
  \wordwrap {
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat.
  }
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\wordwrap-string` *arg* (string)

Wordwrap a string. Paragraphs may be separated with double newlines.

```
\markup {
  \override #'(line-width . 40)
  \wordwrap-string #"Lorem ipsum dolor sit amet, consectetur
    adipisicing elit, sed do eiusmod tempor incididunt ut labore
    et dolore magna aliqua.

    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat.

    Excepteur sint occaecat cupidatat non proident, sunt in culpa
    qui officia deserunt mollit anim id est laborum"
}
```

Lorem ipsum dolor sit amet,  
 consectetur adipisicing elit, sed do  
 eiusmod tempor incididunt ut labore et  
 dolore magna aliqua.  
 Ut enim ad minim veniam, quis  
 nostrud exercitation ullamco laboris  
 nisi ut aliquip ex ea commodo  
 consequat.  
 Excepteur sint occaecat cupidatat non  
 proident, sunt in culpa qui o cia  
 deserunt mollit anim id est laborum

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`
- `baseline-skip`

### A.11.3 Graphic

`\arrow-head` *axis* (integer) *dir* (direction) *filled* (boolean)

Produce an arrow head in specified direction and axis. Use the filled head if *filled* is specified.

```

\markup {
  \fontsize #5 {
    \general-align #Y #DOWN {
      \arrow-head #Y #UP ##t
      \arrow-head #Y #DOWN ##f
      \hspace #2
      \arrow-head #X #RIGHT ##f
      \arrow-head #X #LEFT ##f
    }
  }
}

```

▲ √ > <

`\beam` *width* (number) *slope* (number) *thickness* (number)

Create a beam with the specified parameters.

```

\markup {
  \beam #5 #1 #2
}

```



`\bracket` *arg* (markup)

Draw vertical brackets around *arg*.

```

\markup {
  \bracket {
    \note #"2." #UP
  }
}

```

}

[J.]

`\circle arg` (markup)

Draw a circle around *arg*. Use `thickness`, `circle-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \circle {
    Hi
  }
}
```

Ⓜ

Used properties:

- `circle-padding` (0.2)
- `font-size` (0)
- `thickness` (1)

`\draw-circle radius` (number) *thickness* (number) *filled* (boolean)

A circle of radius *radius* and thickness *thickness*, optionally filled.

```
\markup {
  \draw-circle #2 #0.5 ##f
  \hspace #2
  \draw-circle #2 #0 ##t
}
```

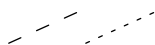


`\draw-dashed-line dest` (pair of numbers)

A dashed line.

If `full-length` is set to `#t` (default) the dashed-line extends to the whole length given by *dest*, without white space at beginning or end. `off` will then be altered to fit. To insist on the given (or default) values of `on`, `off` use `\override #'(full-length . #f)` Manual settings for `on`, `off` and `phase` are possible.

```
\markup {
  \draw-dashed-line #'(5.1 . 2.3)
  \override #'(on . 0.3)
  \override #'(off . 0.5)
  \draw-dashed-line #'(5.1 . 2.3)
}
```



Used properties:

- `full-length` (`#t`)
- `phase` (0)
- `off` (1)

- `on` (1)
- `thickness` (1)

`\draw-dotted-line` *dest* (pair of numbers)

A dotted line.

The dotted-line always extends to the whole length given by *dest*, without white space at beginning or end. Manual settings for `off` are possible to get larger or smaller space between the dots. The given (or default) value of `off` will be altered to fit the line-length.

```
\markup {
  \draw-dotted-line #'(5.1 . 2.3)
  \override #'(thickness . 2)
  \override #'(off . 0.2)
  \draw-dotted-line #'(5.1 . 2.3)
}
```



Used properties:

- `phase` (0)
- `off` (1)
- `thickness` (1)

`\draw-hline`

Draws a line across a page, where the property `span-factor` controls what fraction of the page is taken up.

```
\markup {
  \column {
    \draw-hline
    \override #'(span-factor . 1/3)
    \draw-hline
  }
}
```



Used properties:

- `span-factor` (1)
- `line-width`
- `draw-line-markup`

`\draw-line` *dest* (pair of numbers)

A simple line.

```
\markup {
  \draw-line #'(4 . 4)
  \override #'(thickness . 5)
  \draw-line #'(-3 . 0)
}
```



Used properties:

- **thickness** (1)

`\draw-squiggle-line` *sq-length* (number) *dest* (pair of numbers) *eq-end?* (boolean)

A squiggled line.

If *eq-end?* is set to **#t**, it is ensured the squiggled line ends with a bow in same direction as the starting one. *sq-length* is the length of the first bow. *dest* is the end point of the squiggled line. To match *dest* the squiggled line is scaled accordingly. Its appearance may be customized by overrides for **thickness**, **angularity**, **height** and **orientation**.

```
\markup
\column {
  \draw-squiggle-line #0.5 #'(6 . 0) ##t
  \override #'(orientation . -1)
  \draw-squiggle-line #0.5 #'(6 . 0) ##t
  \draw-squiggle-line #0.5 #'(6 . 0) ##f
  \override #'(height . 1)
  \draw-squiggle-line #0.5 #'(6 . 0) ##t
  \override #'(thickness . 5)
  \draw-squiggle-line #0.5 #'(6 . 0) ##t
  \override #'(angularity . 2)
  \draw-squiggle-line #0.5 #'(6 . 0) ##t
}
```

~~~~~

~~~~~

~~~~~

~~~~~

~~~~~

~~~~~

Used properties:

- **orientation** (1)
- **height** (0.5)
- **angularity** (0)
- **thickness** (0.5)

`\ellipse` *arg* (markup)

Draw an ellipse around *arg*. Use **thickness**, **x-padding**, **y-padding** and **font-size** properties to determine line thickness and padding around the markup.

```
\markup {
  \ellipse {
    Hi
  }
}
```

Ⓜ

Used properties:

- **y-padding** (0.2)
- **x-padding** (0.2)

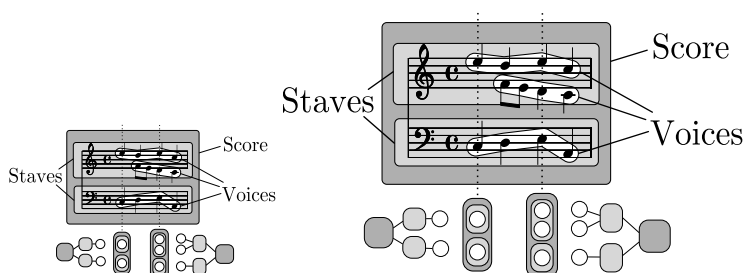


- `font-size` (0)
- `thickness` (1)

`\epsfile axis` (number) `size` (number) `file-name` (string)

Inline an EPS image. The image is scaled along `axis` to `size`.

```
\markup {
  \general-align #Y #DOWN {
    \epsfile #X #20 #"context-example.eps"
    \epsfile #Y #20 #"context-example.eps"
  }
}
```



`\filled-box xext` (pair of numbers) `yext` (pair of numbers) `blot` (number)

Draw a box with rounded corners of dimensions `xext` and `yext`. For example,

```
\filled-box #'(-.3 . 1.8) #'(-.3 . 1.8) #0
```

creates a box extending horizontally from -0.3 to 1.8 and vertically from -0.3 up to 1.8, with corners formed from a circle of diameter 0 (i.e., sharp corners).

```
\markup {
  \filled-box #'(0 . 4) #'(0 . 4) #0
  \filled-box #'(0 . 2) #'(-4 . 2) #0.4
  \filled-box #'(1 . 8) #'(0 . 7) #0.2
  \with-color #white
  \filled-box #'(-4.5 . -2.5) #'(3.5 . 5.5) #0.7
}
```



`\hbracket arg` (markup)

Draw horizontal brackets around `arg`.

```
\markup {
  \hbracket {
    \line {
      one two three
    }
  }
}
```

one two three

`\oval arg (markup)`

Draw an oval around *arg*. Use `thickness`, `x-padding`, `y-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \oval {
    Hi
  }
}
```



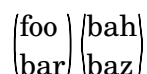
Used properties:

- `y-padding` (0.75)
- `x-padding` (0.75)
- `font-size` (0)
- `thickness` (1)

`\parenthesize arg (markup)`

Draw parentheses around *arg*. This is useful for parenthesizing a column containing several lines of text.

```
\markup {
  \line {
    \parenthesize {
      \column {
        foo
        bar
      }
    }
    \override #'(angularity . 2) {
      \parenthesize {
        \column {
          bah
          baz
        }
      }
    }
  }
}
```



Used properties:

- `width` (0.25)
- `thickness` (1)
- `size` (1)
- `padding`
- `angularity` (0)

`\path` *thickness* (number) *commands* (list)

Draws a path with line *thickness* according to the directions given in *commands*. *commands* is a list of lists where the *car* of each sublist is a drawing command and the *cdr* comprises the associated arguments for each command.

There are seven commands available to use in the list *commands*: `moveto`, `rmoveto`, `lineto`, `rlineto`, `curveto`, `rcurveto`, and `closepath`. Note that the commands that begin with *r* are the relative variants of the other three commands.

The commands `moveto`, `rmoveto`, `lineto`, and `rlineto` take 2 arguments; they are the X and Y coordinates for the destination point.

The commands `curveto` and `rcurveto` create cubic Bézier curves, and take 6 arguments; the first two are the X and Y coordinates for the first control point, the second two are the X and Y coordinates for the second control point, and the last two are the X and Y coordinates for the destination point.

The `closepath` command takes zero arguments and closes the current subpath in the active path.

Note that a sequence of commands *must* begin with a `moveto` or `rmoveto` to work with the SVG output.

Line-cap styles and line-join styles may be customized by overriding the `line-cap-style` and `line-join-style` properties, respectively. Available line-cap styles are 'butt, 'round, and 'square. Available line-join styles are 'miter, 'round, and 'bevel.

The property `filled` specifies whether or not the path is filled with color.

`samplePath =`

```
#'((moveto 0 0)
   (lineto -1 1)
   (lineto 1 1)
   (lineto 1 -1)
   (curveto -5 -5 -5 5 -1 0)
   (closepath))
```

`\markup {`

```
\path #0.25 #samplePath
```

```
\override #'(line-join-style . miter) \path #0.25 #samplePath
```

```
\override #'(filled . #t) \path #0.25 #samplePath
```

`}`



Used properties:

- `filled` (`#f`)
- `line-join-style` (round)
- `line-cap-style` (round)

`\postscript` *str* (string)

This inserts *str* directly into the output as a PostScript command string.

`ringsps = #"`

```
0.15 setlinewidth
```

```

0.9 0.6 moveto
0.4 0.6 0.5 0 361 arc
stroke
1.0 0.6 0.5 0 361 arc
stroke
"

rings = \markup {
  \with-dimensions #'(-0.2 . 1.6) #'(0 . 1.2)
  \postscript #ringsps
}

\relative c'' {
  c2^\rings
  a2_\rings
}

```



`\rounded-box` *arg* (markup)

Draw a box with rounded corners around *arg*. Looks at **thickness**, **box-padding** and **font-size** properties to determine line thickness and padding around the markup; the **corner-radius** property makes it possible to define another shape for the corners (default is 1).

```

c4^\markup {
  \rounded-box {
    Overtura
  }
}
c,8. c16 c4 r

```



Used properties:

- **box-padding** (0.5)
- **font-size** (0)
- **corner-radius** (1)
- **thickness** (1)

`\scale` *factor-pair* (pair of numbers) *arg* (markup)

Scale *arg*. *factor-pair* is a pair of numbers representing the scaling-factor in the X and Y axes. Negative values may be used to produce mirror images.

```

\markup {
  \line {
    \scale #'(2 . 1)
    stretched
    \scale #'(1 . -1)
  }
}

```

```

        mirrored
    }
}

```

**stretched** 

`\triangle` *filled* (boolean)

A triangle, either filled or empty.

```

\markup {
  \triangle ##t
  \hspace #2
  \triangle ##f
}

```



Used properties:

- `baseline-skip` (2)
- `font-size` (0)
- `thickness` (0.1)

`\with-url` *url* (string) *arg* (markup)

Add a link to URL *url* around *arg*. This only works in the PDF backend.

```

\markup {
  \with-url #"http://lilypond.org/" {
    LilyPond ... \italic {
      music notation for everyone
    }
  }
}

```

LilyPond ... *music notation for everyone*

#### A.11.4 Music

`\compound-meter` *time-sig* (number or pair)

Draw a numeric time signature.

```

\markup {
  \column {
    \line { Single number: \compound-meter #3 }
    \line { Conventional: \compound-meter #'(4 . 4)
              or \compound-meter #'(4 4) }
    \line { Compound: \compound-meter #'(2 3 8) }
    \line { Single-number compound: \compound-meter #'((2) (3)) }
    \line { Complex compound: \compound-meter #'((2 3 8) (3 4)) }
  }
}

```

Single number: **3**  
 Conventional:  **$\frac{4}{4}$**  or  **$\frac{4}{4}$**   
 Compound:  **$\frac{2+3}{8}$**   
 Single-number compound:  **$\frac{2+3}{8}$**   
 Complex compound:  **$\frac{2+3+3}{8}$**

`\customTabClef` *num-strings* (integer) *staff-space* (number)

Draw a tab clef sans-serif style.

`\doubleflat`

Draw a double flat symbol.

```
\markup {
  \doubleflat
}
```

**bb**

`\doublesharp`

Draw a double sharp symbol.

```
\markup {
  \doublesharp
}
```

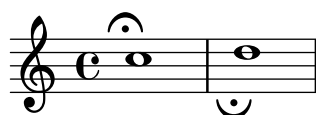
**##**

`\fermata`

Create a fermata glyph. When *direction* is DOWN, use an inverted glyph. Note that within music, one would usually use the `\fermata` articulation instead of a markup.

```
{ c''1^ \markup \fermata d''1_ \markup \fermata }
```

```
\markup { \fermata \override #` (direction . ,DOWN) \fermata }
```



Used properties:

- `direction` (1)

`\flat`

Draw a flat symbol.

```
\markup {
  \flat
}
```

**b**

`\musicglyph` *glyph-name* (string)

*glyph-name* is converted to a musical symbol; for example, `\musicglyph # "accidentals.natural"` selects the natural sign from the music font. See Sezione

“The Feta font” in *Guida alla Notazione* for a complete listing of the possible glyphs.

```
\markup {
  \musicglyph #"f"
  \musicglyph #"rests.2"
  \musicglyph #"clefs.G_change"
}
```



`\natural`

Draw a natural symbol.

```
\markup {
  \natural
}
```



`\note-by-number` *log* (number) *dot-count* (number) *dir* (number)

Construct a note symbol, with stem and flag. By using fractional values for *dir*, longer or shorter stems can be obtained. Supports all note-head-styles. Ancient note-head-styles will get mensural-style-flags. **flag-style** may be overridden independently. Supported flag-styles are **default**, **old-straight-flag**, **modern-straight-flag**, **flat-flag**, **mensural** and **neomensural**. The latter two flag-styles will both result in mensural-flags. Both are supplied for convenience.

```
\markup {
  \note-by-number #3 #0 #DOWN
  \hspace #2
  \note-by-number #1 #2 #0.8
}
```



Used properties:

- **style** ('())
- **flag-style** ('())
- **font-size** (0)

`\note` *duration* (string) *dir* (number)

This produces a note with a stem pointing in *dir* direction, with the *duration* for the note head type and augmentation dots. For example, `\note #"4." #-0.75` creates a dotted quarter note, with a shortened down stem.

```
\markup {
  \override #'(style . cross) {
    \note #"4.." #UP
  }
  \hspace #2
  \note #"breve" #0
}
```

}

↓  
x.. 

Used properties:

- `style '()`
- `flag-style '()`
- `font-size (0)`

`\rest-by-number` *log* (number) *dot-count* (number)

A rest or multi-measure-rest symbol.

```
\markup {
  \rest-by-number #3 #2
  \hspace #2
  \rest-by-number #0 #1
  \hspace #2
  \override #'(multi-measure-rest . #t)
  \rest-by-number #0 #0
}
```

γ..  

Used properties:





- `multi-measure-rest (#f)`
- `style '()`
- `font-size (0)`

`\rest` *duration* (string)

This produces a rest, with the *duration* for the rest type and augmentation dots. "breve", "longa" and "maxima" are valid input-strings.

Printing MultiMeasureRests could be enabled with `\override #'(multi-measure-rest . #t)` If MultiMeasureRests are taken, the MultiMeasureRestNumber is printed above. This is enabled for all styles using default-glyphs. Could be disabled with `\override #'(multi-measure-rest-number . #f)`

```
\markup {
  \rest #"4.."
  \hspace #2
  \rest #"breve"
  \hspace #2
  \override #'(multi-measure-rest . #t)
  {
    \rest #"7"
    \hspace #2
    \override #'(multi-measure-rest-number . #f)
    \rest #"7"
  }
}
```

ζ..    

Used properties:

- `word-space (0.6)`



- `multi-measure-rest-number (#t)`
- `multi-measure-rest (#f)`
- `style ('())`

`\score score (score)`

Inline an image of music. The reference point (usually the middle staff line) of the lowest staff in the top system is placed on the baseline.

```
\markup {
  \score {
    \new PianoStaff <<
      \new Staff \relative c' {
        \key f \major
        \time 3/4
        \mark \markup { Allegro }
        f2\p( a4)
        c2( a4)
        bes2( g'4)
        f8( e) e4 r
      }
      \new Staff \relative c {
        \clef bass
        \key f \major
        \time 3/4
        f8( a c a c a
        f c' es c es c)
        f,( bes d bes d bes)
        f( g bes g bes g)
      }
    >>
  }
  \layout {
    indent = 0.0\cm
    \context {
      \Score
      \override RehearsalMark
        #'break-align-symbols = #'(time-signature key-signature)
      \override RehearsalMark
        #'self-alignment-X = #LEFT
    }
    \context {
      \Staff
      \override TimeSignature
        #'break-align-anchor-alignment = #LEFT
    }
  }
}
```



Used properties:

- `baseline-skip`

`\semiflat`

Draw a semiflat symbol.

```
\markup {
  \semiflat
}
```

♭

`\semisharp`

Draw a semisharp symbol.

```
\markup {
  \semisharp
}
```

♮

`\sesquiflat`

Draw a 3/2 flat symbol.

```
\markup {
  \sesquiflat
}
```

♭

`\sesquisharp`

Draw a 3/2 sharp symbol.

```
\markup {
  \sesquisharp
}
```

♯

`\sharp`

Draw a sharp symbol.

```
\markup {
  \sharp
}
```

♯

`\tied-lyric str` (string)

Like simple-markup, but use tie characters for ‘~’ tilde symbols.

```
\markup \column {
```

```
\tied-lyric #"Siam navi~all'onde~algenti Lasciate~in abbandono"
\tied-lyric #"Impetuosi venti I nostri~affetti sono"
\tied-lyric #"Ogni diletto~e scoglio Tutta la vita~e~un mar."
}
```

Siam navi~all'onde~algenti Lasciate~in abbandono  
 Impetuosi venti I nostri~a etti sono  
 Ogni diletto~e scoglio Tutta la vita~e un mar.

Used properties:

- `word-space`

### A.11.5 Instrument Specific Markup

`\fret-diagram` *definition-string* (string)

Make a (guitar) fret diagram. For example, say

```
\markup \fret-diagram #"s:0.75;6-x;5-x;4-o;3-2;2-3;1-2;"
```

for fret spacing 3/4 of staff space, D chord diagram

Syntax rules for *definition-string*:

- Diagram items are separated by semicolons.
- Possible items:
  - `s: number` – Set the fret spacing of the diagram (in staff spaces). Default: 1.
  - `t: number` – Set the line thickness (relative to normal line thickness). Default: 0.5.
  - `h: number` – Set the height of the diagram in frets. Default: 4.
  - `w: number` – Set the width of the diagram in strings. Default: 6.
  - `f: number` – Set fingering label type (0 = none, 1 = in circle on string, 2 = below string). Default: 0.
  - `d: number` – Set radius of dot, in terms of fret spacing. Default: 0.25.
  - `p: number` – Set the position of the dot in the fret space. 0.5 is centered; 1 is on lower fret bar, 0 is on upper fret bar. Default: 0.6.
  - `c: string1-string2-fret` – Include a barre mark from *string1* to *string2* on *fret*.
  - `string-fret` – Place a dot on *string* at *fret*. If *fret* is ‘o’, *string* is identified as open. If *fret* is ‘x’, *string* is identified as muted.
  - `string-fret-fingering` – Place a dot on *string* at *fret*, and label with *fingering* as defined by the `f:` code.
- Note: There is no limit to the number of fret indications per string.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\fret-diagram-terse` *definition-string* (string)

Make a fret diagram markup using terse string-based syntax.

Here is an example

```
\markup \fret-diagram-terse #"x;x;o;2;3;2;"
```

for a D chord diagram.

Syntax rules for *definition-string*:

- Strings are terminated by semicolons; the number of semicolons is the number of strings in the diagram.
- Mute strings are indicated by ‘x’.
- Open strings are indicated by ‘o’.
- A number indicates a fret indication at that fret.
- If there are multiple fret indicators desired on a string, they should be separated by spaces.
- Fingerings are given by following the fret number with a -, followed by the finger indicator, e.g. ‘3-2’ for playing the third fret with the second finger.
- Where a barre indicator is desired, follow the fret (or fingering) symbol with -( to start a barre and -) to end the barre.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\fret-diagram-verbose` *marking-list* (pair)

Make a fret diagram containing the symbols indicated in *marking-list*.

For example,

```
\markup \fret-diagram-verbose
#'(mute 6) (mute 5) (open 4)
  (place-fret 3 2) (place-fret 2 3) (place-fret 1 2))
```

produces a standard D chord diagram without fingering indications.

Possible elements in *marking-list*:

`(mute string-number)`

Place a small ‘x’ at the top of string *string-number*.

`(open string-number)`

Place a small ‘o’ at the top of string *string-number*.

`(barre start-string end-string fret-number)`

Place a barre indicator (much like a tie) from string *start-string* to string *end-string* at fret *fret-number*.

`(capo fret-number)`

Place a capo indicator (a large solid bar) across the entire fretboard at fret location *fret-number*. Also, set fret *fret-number* to be the lowest fret on the fret diagram.

`(place-fret string-number fret-number [finger-value] [color-modifier] [color] ['parenthesized ['default-paren-color]])` Place a fret playing indication on string *string-number* at fret *fret-number* with an optional fingering label *finger-value*, an optional color modifier *color-modifier*, an optional color *color*, an optional parenthesis 'parenthesized and an optional parenthesis color 'default-paren-color. By default, the fret playing indicator is a solid dot. This can be globally changed by setting the value of

the variable *dot-color* or for a single dot by setting the value of *color*. The dot can be parenthesized by adding '**parenthesized**'. By default the color for the parenthesis is taken from the dot. Adding '**default-paren-color**' will take the parenthesis-color from the global *dot-color*, as a fall-back black will be used. Setting *color-modifier* to **inverted** inverts the dot color for a specific fingering. The values for *string-number*, *fret-number*, and the optional *finger* should be entered first in that order. The order of the other optional arguments does not matter. If the *finger* part of the **place-fret** element is present, *finger-value* will be displayed according to the setting of the variable *finger-code*. There is no limit to the number of fret indications per string.

Used properties:

- **thickness** (0.5)
- **fret-diagram-details**
- **size** (1.0)
- **align-dir** (-0.4)

**\harp-pedal** *definition-string* (string)

Make a harp pedal diagram.

Possible elements in *definition-string*:

- ^           pedal is up
- pedal is neutral
- v           pedal is down
- |           vertical divider line
- o           the following pedal should be circled (indicating a change)

The function also checks if the string has the typical form of three pedals, then the divider and then the remaining four pedals. If not it prints out a warning. However, in any case, it will also print each symbol in the order as given. This means you can place the divider (even multiple dividers) anywhere you want, but you'll have to live with the warnings.

The appearance of the diagram can be tweaked inter alia using the size property of the TextScript grob (**\override Voice.TextScript #'size = #0.3**) for the overall, the thickness property (**\override Voice.TextScript #'thickness = #3**) for the line thickness of the horizontal line and the divider. The remaining configuration (box sizes, offsets and spaces) is done by the **harp-pedal-details** list of properties (**\override Voice.TextScript #'harp-pedal-details #'box-width = #1**). It contains the following settings: **box-offset** (vertical shift of the box center for up/down pedals), **box-width**, **box-height**, **space-before-divider** (the spacing between two boxes before the divider) and **space-after-divider** (box spacing after the divider).

**\markup \harp-pedal #"^-v|--ov^"**



Used properties:

- **thickness** (0.5)

- `harp-pedal-details` `('')`
- `size` `(1.2)`

`\woodwind-diagram` *instrument* (symbol) *user-draw-commands* (list)

Make a woodwind-instrument diagram. For example, say

```
\markup \woodwind-diagram
  #'oboe #'((lh . (d ees)) (cc . (five3qT1q)) (rh . (gis)))
```

for an oboe with the left-hand d key, left-hand ees key, and right-hand gis key depressed while the five-hole of the central column effectuates a trill between 1/4 and 3/4 closed.

The following instruments are supported:

- piccolo
- flute
- oboe
- clarinet
- bass-clarinet
- saxophone
- bassoon
- contrabassoon

To see all of the callable keys for a given instrument, include the function (`print-keys 'instrument`) in your `.ly` file, where `instrument` is the instrument whose keys you want to print.

Certain keys allow for special configurations. The entire gamut of configurations possible is as follows:

- `1q` (1/4 covered)
- `1h` (1/2 covered)
- `3q` (3/4 covered)
- `R` (ring depressed)
- `F` (fully covered; the default if no state put)

Additionally, these configurations can be used in trills. So, for example, `three3qTR` effectuates a trill between 3/4 full and ring depressed on the three hole. As another example, `threeRT` effectuates a trill between `R` and open, whereas `threeTR` effectuates a trill between open and shut. To see all of the possibilities for all of the keys of a given instrument, invoke (`print-keys-verbose 'instrument`).

Lastly, substituting an empty list for the pressed-key alist will result in a diagram with all of the keys drawn but none filled, for example:

```
\markup \woodwind-diagram #'oboe #()
```

Used properties:

- `graphical` `(#t)`
- `thickness` `(0.1)`
- `size` `(1)`

### A.11.6 Accordion Registers

`\discant` *name* (string)

`\discant` *name* generates a discant accordion register symbol.

To make it available,





```
#(use-modules (scm accreg))
```

is required near the top of your input file.

The register names in the default `\discant` register set have modeled after numeric Swiss notation like depicted in [http://de.wikipedia.org/wiki/Register\\_%28Akkordeon%29](http://de.wikipedia.org/wiki/Register_%28Akkordeon%29), omitting the slashes and dropping leading zeros.

The string *name* is basically a three-digit number with the lowest digit specifying the number of 16' reeds, the tens the number of 8' reeds, and the hundreds specifying the number of 4' reeds. Without modification, the specified number of reeds in 8' is centered in the symbol. Newer instruments may have registrations where 8' can be used either within or without a tone chamber, 'cassotto'. Notationally, the central dot then indicates use of cassotto. One can suffix the tens' digits '1' and '2' with '+' or '-' to indicate clustering the dots at the right or left respectively rather than centered.

Some examples are

	
<code>\discant #"1"</code>	<code>\discant #"1+0"</code>
	
<code>\discant #"120"</code>	<code>\discant #"131"</code>

Used properties:

- `font-size` (0)

`\freeBass` *name* (string)




`\freeBass` *name* generates a free bass/converter accordion register symbol for the usual two-reed layout.

To make it available,

```
#(use-modules (scm accreg))
```

is required near the top of your input file.

Available registrations are

	
<code>\freeBass #"1"</code>	<code>\freeBass #"11"</code>
	
<code>\freeBass #"10"</code>	

Used properties:

- `font-size` (0)

`\stdBass` *name* (string)

`\stdBass` *name* generates a standard bass accordion register symbol.

To make it available,

```
#(use-modules (scm accreg))
```

is required near the top of your input file.

The default bass register definitions have been modeled after the article <http://www.accordions.com/index/art/stradella.shtml> originally appearing in Accord Magazine.








The underlying register model is



This kind of overlapping arrangement is common for Italian instruments though the exact location of the octave breaks differ.

When not composing for a particular target instrument, using the five reed definitions makes more sense than using a four reed layout: in that manner, the ‘**Master**’ register is unambiguous. This is rather the rule in literature bothering about bass registrations at all.

Available registrations are

	
<code>\stdBass #"Soprano"</code>	<code>\stdBass #"Soft Bass"</code>
	
<code>\stdBass #"Alto"</code>	<code>\stdBass #"Soft Tenor"</code>
	
<code>\stdBass #"Tenor"</code>	<code>\stdBass #"Bass/Alto"</code>
	
<code>\stdBass #"Master"</code>	

Used properties:

- `font-size (0)`

`\stdBassIV` *name* (string)

`\stdBassIV` *name* generates a standard bass accordion register symbol.

To make it available,

`#(use-modules (scm accreg))`

is required near the top of your input file.

The main use is for four-reed standard bass instruments with reedbank layout













Notable instruments are Morino models with MIII (the others are five-reed instead) and the Atlantic IV. Most of those models have three register switches. Some newer Morinos with MIII might have five or even seven.

The prevalent three-register layout uses the middle three switches ‘**Tenor**’, ‘**Master**’, ‘**Soft Bass**’. Note that the sound is quite darker than the same registrations of ‘**c**,’-based instruments.

Available registrations are

	
<code>\stdBassIV #"Soprano"</code>	<code>\stdBassIV #"Soft Bass"</code>
	
<code>\stdBassIV #"Alto"</code>	<code>\stdBassIV #"Bass/Alto"</code>
	
<code>\stdBassIV #"Tenor"</code>	<code>\stdBassIV #"Soft Bass/Alto"</code>
	
<code>\stdBassIV #"Master"</code>	<code>\stdBassIV #"Soft Tenor"</code>

Used properties:

- `font-size (0)`

`\stdBassV` *name* (string)

`\stdBassV` *name* generates a standard bass accordion register symbol.

To make it available,

`#(use-modules (scm accreg))`

is required near the top of your input file.

The main use is for five-reed standard bass instruments with reedbank layout













This tends to be the bass layout for Hohner’s Morino series without convertor or MIII manual.

With the exception of the rather new 7-register layout, the highest two chord reeds are usually sounded together. The Older instruments offer 5 or 3 bass registers. The Tango VM offers an additional ‘Solo Bass’ setting that mutes the chord reeds. The symbol on the register buttons of the Tango VM would actually match the physical five-octave layout reflected here, but it is not used in literature.

Composers should likely prefer the five-reed versions of these symbols. The mismatch of a four-reed instrument with five-reed symbols is easier to resolve for the player than the other way round.

Available registrations are

	
<code>\stdBassV #"Bass/Alto"</code>	<code>\stdBassV #"Soft Bass"</code>
	
<code>\stdBassV #"Soft Bass/Alto"</code>	<code>\stdBassV #"Soft Tenor"</code>
	
<code>\stdBassV #"Alto"</code>	<code>\stdBassV #"Soprano"</code>
	
<code>\stdBassV #"Tenor"</code>	<code>\stdBassV #"Sopranos"</code>
	
<code>\stdBassV #"Master"</code>	<code>\stdBassV #"Solo Bass"</code>

Used properties:

- `font-size (0)`

`\stdBassVI` *name* (string)

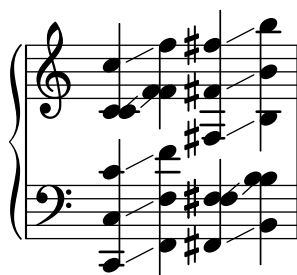
`\stdBassVI` *name* generates a standard bass accordion register symbol for six reed basses.

To make it available,

`#(use-modules (scm accreg))`

is required near the top of your input file.








This is primarily the register layout for the Hohner “Gola” model. The layout is



The registers are effectively quite similar to that of `\stdBass`. An additional bass reed at alto pitch is omitted for esthetical reasons from the ‘Master’ setting, so the

symbols are almost the same except for the ‘Alto/Soprano’ register with bass notes at Alto pitch and chords at Soprano pitch.

Available registrations are

	
<code>\stdBassVI #"Soprano"</code>	<code>\stdBassVI #"Alto/Soprano"</code>
	
<code>\stdBassVI #"Alto"</code>	<code>\stdBassVI #"Bass/Alto"</code>
	
<code>\stdBassVI #"Soft Tenor"</code>	<code>\stdBassVI #"Soft Bass"</code>
	
<code>\stdBassVI #"Master"</code>	

Used properties:

- `font-size` (0)

### A.11.7 Other

`\auto-footnote mkup (markup) note (markup)`

Have footnote *note* act as an annotation to the markup *mkup*.

```
\markup {
  \auto-footnote a b
  \override #'(padding . 0.2)
  \auto-footnote c d
}
```

**a c**

The footnote will be annotated automatically.

Used properties:

- `padding` (0.0)
- `raise` (0.5)

`\backslashed-digit num (integer)`

A feta number, with backslash. This is for use in the context of figured bass notation.

```
\markup {
  \backslashed-digit #5
  \hspace #2
  \override #'(thickness . 3)
  \backslashed-digit #7
}
```

**5 7**

Used properties:

- `thickness` (1.6)

- `font-size (0)`

`\char num` (integer)

Produce a single character. Characters encoded in hexadecimal format require the prefix `#x`.

```
\markup {
  \char #65 \char ##x00a9
}
```

A ©

`\eyeglasses`

Prints out eyeglasses, indicating strongly to look at the conductor.

```
\markup { \eyeglasses }
```



`\first-visible args` (markup list)

Use the first markup in *args* that yields a non-empty stencil and ignore the rest.

```
\markup {
  \first-visible {
    \fromproperty #'header:composer
    \italic Unknown
  }
}
```

*Unknown*

`\footnote mkup` (markup) *note* (markup)

Have footnote *note* act as an annotation to the markup *mkup*.

```
\markup {
  \auto-footnote a b
  \override #'(padding . 0.2)
  \auto-footnote c d
}
```

a c

The footnote will not be annotated automatically.

`\fraction arg1` (markup) *arg2* (markup)

Make a fraction of two markups.

```
\markup {
  π ≈
  \fraction 355 113
}
```

$\pi \approx \frac{355}{113}$

Used properties:

- `font-size (0)`

`\fromproperty symbol (symbol)`

Read the *symbol* from property settings, and produce a stencil from the markup contained within. If *symbol* is not defined, it returns an empty markup.

```
\header {
  myTitle = "myTitle"
  title = \markup {
    from
    \italic
    \fromproperty #'header:myTitle
  }
}
\markup {
  \null
}
```

**from *myTitle***

`\left-brace size (number)`

A feta brace in point size *size*.

```
\markup {
  \left-brace #35
  \hspace #2
  \left-brace #45
}
```

{ }

`\lookup glyph-name (string)`

Lookup a glyph by name.

```
\markup {
  \override #'(font-encoding . fetaBraces) {
    \lookup #"brace200"
    \hspace #2
    \rotate #180
    \lookup #"brace180"
  }
}
```

{ }

`\markalphabet` *num* (integer)

Make a markup letter for *num*. The letters start with A to Z and continue with double letters.

```
\markup {
  \markalphabet #8
  \hspace #2
  \markalphabet #26
}
```

I AA

`\markletter` *num* (integer)

Make a markup letter for *num*. The letters start with A to Z (skipping letter I), and continue with double letters.

```
\markup {
  \markletter #8
  \hspace #2
  \markletter #26
}
```

J AB

`\null`

An empty markup with extents of a single point.

```
\markup {
  \null
}
```

`\on-the-fly` *procedure* (procedure) *arg* (markup)

Apply the *procedure* markup command to *arg*. *procedure* takes the same arguments as `interpret-markup` and returns a stencil.

`\override` *new-prop* (pair) *arg* (markup)

Add the argument *new-prop* to the property list. Properties may be any property supported by Sezione “font-interface” in *Guida al Funzionamento Interno*, Sezione “text-interface” in *Guida al Funzionamento Interno* and Sezione “instrument-specific-markup-interface” in *Guida al Funzionamento Interno*.

```
\markup {
  \line {
    \column {
      default
      baseline-skip
    }
    \hspace #2
    \override #'(baseline-skip . 4) {
      \column {
        increased
        baseline-skip
      }
    }
  }
}
```

```

    }
  }

  default      increased
  baseline-skip baseline-skip

```

`\page-link` *page-number* (number) *arg* (markup)  
 Add a link to the page *page-number* around *arg*. This only works in the PDF backend.

```

\markup {
  \page-link #2 { \italic { This links to page 2... } }
}

```

*This links to page 2...*

`\page-ref` *label* (symbol) *gauge* (markup) *default* (markup)  
 Reference to a page number. *label* is the label set on the referenced page (using the `\label` command), *gauge* a markup used to estimate the maximum width of the page number, and *default* the value to display when *label* is not found.  
 (If the current book or bookpart is set to use roman numerals for page numbers, the reference will be formatted accordingly – in which case the *gauge*'s width may require additional tweaking.)

`\pattern` *count* (integer) *axis* (integer) *space* (number) *pattern* (markup)  
 Prints *count* times a *pattern* markup. Patterns are spaced apart by *space*. Patterns are distributed on *axis*.

```

\markup \column {
  "Horizontally repeated : "
  \pattern #7 #X #2 \flat
  \null
  "Vertically repeated : "
  \pattern #3 #Y #0.5 \flat
}

```

Horizontally repeated :

b b b b b b b

Vertically repeated :

b  
 b  
 b

`\property-recursive` *symbol* (symbol)  
 Print out a warning when a header field markup contains some recursive markup definition.

`\right-brace` *size* (number)  
 A feta brace in point size *size*, rotated 180 degrees.

```

\markup {
  \right-brace #45
  \hspace #2
}

```

```
\right-brace #35
}
```

```
{ {
```

`\slashed-digit` *num* (integer)

A feta number, with slash. This is for use in the context of figured bass notation.

```
\markup {
  \slashed-digit #5
  \hspace #2
  \override #'(thickness . 3)
  \slashed-digit #7
}
```

**5 7**

Used properties:

- `thickness` (1.6)
- `font-size` (0)

`\stencil` *stil* (stencil)

Use a stencil as markup.

```
\markup {
  \stencil #(make-circle-stencil 2 0 #t)
}
```



`\strut`

Create a box of the same height as the space in the current font.

`\transparent` *arg* (markup)

Make *arg* transparent.

```
\markup {
  \transparent {
    invisible text
  }
}
```

`\verbatim-file` *name* (string)

Read the contents of file *name*, and include it verbatim.

```
\markup {
  \verbatim-file #"simple.ly"
}
```



```

%% A simple piece in LilyPond, a scale.
\relative {
  c' d e f g a b c
}
%% Optional helper for automatic updating by convert-ly.
%% May be omitted.
\version "2.19.21"

```

`\whiteout` *arg* (markup)

Provide a white background for *arg*. The shape of the white background is determined by *style*. The default is `box` which produces a rectangle. `rounded-box` produces a rounded rectangle. `outline` approximates the outline of the markup.

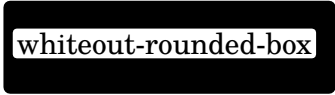
```

\markup {
  \combine
    \filled-box #'(-1 . 15) #'(-3 . 4) #1
    \override #'(thickness . 1.5)
    \whiteout whiteout-box
}
\markup {
  \combine
    \filled-box #'(-1 . 24) #'(-3 . 4) #1
    \override #'(style . rounded-box)
    \override #'(thickness . 3)
    \whiteout whiteout-rounded-box
}
\markup {
  \combine
    \filled-box #'(-1 . 18) #'(-3 . 4) #1
    \override #'(style . outline)
    \override #'(thickness . 3)
    \whiteout whiteout-outline
}

```



whiteout-box



whiteout-rounded-box



whiteout-outline

Used properties:

- `thickness` ('())
- `style` (box)

`\with-color` *color* (color) *arg* (markup)

Draw *arg* in color specified by *color*.

```

\markup {
  \with-color #red
  red
}

```

```

\hspace #2
\with-color #green
green
\hspace #2
\with-color #blue
blue
}

```

**red green blue**

`\with-dimensions` *x* (pair of numbers) *y* (pair of numbers) *arg* (markup)

Set the dimensions of *arg* to *x* and *y*.

`\with-link` *label* (symbol) *arg* (markup)

Add a link to the page holding label *label* around *arg*. This only works in the PDF backend.

```

\markup {
  \with-link #'label {
    \italic { This links to the page containing the label... }
  }
}

```

*This links to the page containing the label...*

## A.12 Comandi per una lista di *markup*

Tutti i comandi seguenti possono essere usati all'interno di `\markuplist`:

`\column-lines` *args* (markup list)

Like `\column`, but return a list of lines instead of a single markup. `baseline-skip` determines the space between each markup in *args*.

Used properties:

- `baseline-skip`

`\justified-lines` *args* (markup list)

Like `\justify`, but return a list of lines instead of a single markup. Use `\override-lines #'(line-width . X)` to set the line width; *X* is the number of staff spaces.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\map-markup-commands` *compose* (procedure) *args* (markup list)

This applies the function *compose* to every markup in *args* (including elements of markup list command calls) in order to produce a new markup list. Since the return value from a markup list command call is not a markup list but rather a list of stencils, this requires passing those stencils off as the results of individual markup calls. That way, the results should work out as long as no markups rely on side effects.

`\override-lines` *new-prop* (pair) *args* (markup list)

Like `\override`, for markup lists.

`\score-lines` *score* (*score*)

This is the same as the `\score` markup but delivers its systems as a list of lines. Its *score* argument is entered in braces like it would be for `\score`.

`\table-of-contents`

`\wordwrap-internal` *justify* (boolean) *args* (markup list)

Internal markup list command used to define `\justify` and `\wordwrap`.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)

`\wordwrap-lines` *args* (markup list)

Like `\wordwrap`, but return a list of lines instead of a single markup. Use `\override-lines #'(line-width . X)` to set the line width, where *X* is the number of staff spaces.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\wordwrap-string-internal` *justify* (boolean) *arg* (string)

Internal markup list command used to define `\justify-string` and `\wordwrap-string`.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`

## A.13 Elenco dei caratteri speciali

Si possono usare i seguenti riferimenti per i caratteri speciali; maggiori informazioni in `<undefined>` [ASCII aliases], pagina `<undefined>`.

Si usa la sintassi HTML. Di questi riferimenti molti sono identici a quelli HTML, alcuni sono ispirati a  $\text{\LaTeX}$ .

I caratteri sono compresi in un riquadro per rendere visibile la loro dimensione. È stato aggiunto un po' di spazio tra il carattere e il riquadro per migliorare la leggibilità.

<code>&amp;hellip;</code>	<code>&amp;ndash;</code>	<code>&amp;mdash;</code>	<code>&amp;iexcl;</code>	<code>&amp;quest;</code>	<code>&amp;solidus;</code>	<code>&amp;q;</code>	<code>&amp;frq;</code>
<code>&amp;qq;</code>	<code>&amp;frqq;</code>	<code>&amp;glq;</code>	<code>&amp;grq;</code>	<code>&amp;glqq;</code>	<code>&amp;grqq;</code>	<code>&amp;elq;</code>	<code>&amp;erq;</code>
<code>&amp;elqq;</code>	<code>&amp;erqq;</code>	<code>&amp;ensp;</code>	<code>&amp;emsp;</code>				

<code>&amp;thinsp;</code>	<code>&amp;nbsp;</code>	<code>&amp;nnbsp;</code>	<code>&amp;zwj;</code>	<code>◦</code>
<code>&amp;zwjn;</code>	<code>◦</code>	<code>&amp;middot;</code>	<code>▣</code>	<code>&amp;copyright;</code>
<code>&amp;registered;</code>	<code>®</code>	<code>&amp;trademark;</code>	<code>™</code>	<code>&amp;dagger;</code>
<code>&amp;numero;</code>	<code>№</code>	<code>&amp;ordf;</code>	<code>ª</code>	<code>&amp;para;</code>
<code>&amp;sect;</code>	<code>§</code>	<code>&amp;deg;</code>	<code>◻</code>	<code>&amp;numero;</code>
<code>&amp;brvbar;</code>	<code>̄</code>	<code>&amp;acute;</code>	<code>◊</code>	<code>&amp;grave;</code>
<code>&amp;breve;</code>	<code>˘</code>	<code>&amp;caron;</code>	<code>˘</code>	<code>&amp;cedilla;</code>
<code>&amp;diaeresis;</code>	<code>¨</code>	<code>&amp;macron;</code>	<code>ˉ</code>	<code>&amp;circum ex;</code>
<code>&amp;ae;</code>	<code>æ</code>	<code>&amp;AE;</code>	<code>Æ</code>	<code>&amp;dh;</code>
<code>&amp;dj;</code>	<code>đ</code>	<code>&amp;DJ;</code>	<code>Đ</code>	<code>&amp;l;</code>
<code>&amp;ng;</code>	<code>ŋ</code>	<code>&amp;NG;</code>	<code>Ŋ</code>	<code>&amp;o;</code>
<code>&amp;oe;</code>	<code>œ</code>	<code>&amp;OE;</code>	<code>Œ</code>	<code>&amp;s;</code>
<code>&amp;th;</code>	<code>þ</code>	<code>&amp;TH;</code>	<code>Þ</code>	<code>&amp;plus;</code>
<code>&amp;times;</code>	<code>×</code>	<code>&amp;div;</code>	<code>÷</code>	<code>&amp;sup1;</code>
<code>&amp;sup3;</code>	<code>³</code>	<code>&amp;sqrt;</code>	<code>√</code>	<code>&amp;increment;</code>
<code>&amp;sum;</code>	<code>Σ</code>	<code>&amp;pm;</code>	<code>±</code>	<code>&amp;bulletop;</code>
<code>&amp;neg;</code>	<code>¬</code>	<code>&amp;currency;</code>	<code>¤</code>	<code>&amp;dollar;</code>
<code>&amp;pounds;</code>	<code>£</code>	<code>&amp;yen;</code>	<code>¥</code>	<code>&amp;cent;</code>
			<code>¢</code>	
			<code>€</code>	
			<code>€</code>	

## A.14 Elenco delle articolazioni

Le liste seguenti mostrano tutti i segni del tipo di carattere Feta che possono essere attaccati alle note (es: ‘f\accent’ o ‘f->’). Ogni esempio mostra il segno nelle posizioni *up*, *down* e *neutral*.

### Articolazioni

\accent or ->



\espressivo



\marcato or -^



\portato or -\_



\staccatissimo  
or -!



\staccato or -.



\tenuto or --



### Ornamenti

\prall



\prallup



\pralldown



\upprall



\downprall



\prallprall



\lineprall



\prallmordent



\mordent



\upmordent



\downmordent



\trill



\turn



\reverseturn



### Punti coronati

`\shortfermata`



`\fermata`



`\longfermata`



`\verylongfermata`



## Segni specifici per strumento

`\upbow`



`\downbow`



`\flageolet`



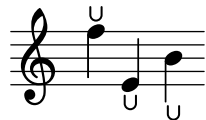
`\open`



`\halfopen`



`\lheel`



`\rheel`



`\ltoe`



`\rtoe`



`\snappizzicato`



`\stopped or -+`



## Segni di ripetizione

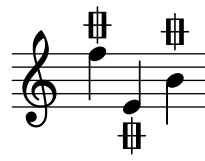
`\segno`



`\coda`

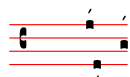


`\varcoda`



## Segni antichi

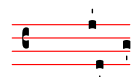
`\accentus`



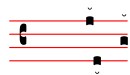
`\circulus`



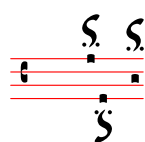
`\ictus`



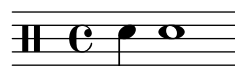
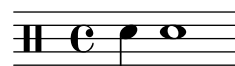
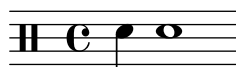
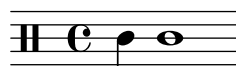
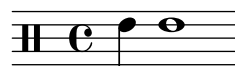
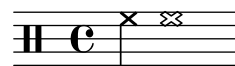
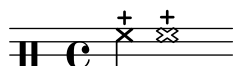
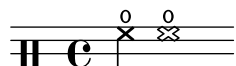
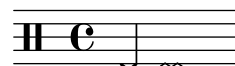
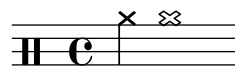
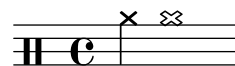
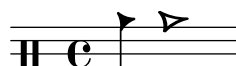
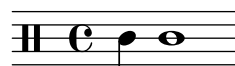
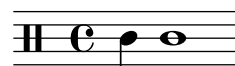
`\semicirculus`



`\signumcongruentiae`



## A.15 Note percussive

bassdrum  
bdacousticbassdrum  
bdasnare  
snacousticsnare  
snaelectricsnare  
snelowfloortom  
tomflhighfloortom  
tomfhlowtom  
tomlhightom  
tomhlowmidtom  
tommlhighmidtom  
tommhhighhat  
hhclosedhihat  
hhcopenhighhat  
hhohalfopenhihat  
hhhopedalhihat  
hhpcrashcymbal  
cymccrashcymbala  
cymcacrashcymbalb  
cymcbridecymbal  
cymrridecymbala  
cymraridecymbalb  
cymrbchinese cymbal  
cymchsplashcymbal  
cymsridebell  
rbcowbell  
cbhibongo  
bohopenhibongo  
boho

mutehibongo  
boh



lobongo  
bol



openlobongo  
bolo



mutelobongo  
bolm



hiconga  
cgh



openhiconga  
cgho



mutehiconga  
cghm



locongga  
cgl



openlocongga  
cglo



mutelocongga  
cglm



hitimbale  
timh



lotimbale  
timl



hiagogo  
agh



loagogo  
agl



sidestick  
ss



hisidestick  
ssh



losidestick  
ssl



guiro  
gui



shortguiro  
guis



longguiro  
guil



cabasa  
cab



maracas  
mar



shortwhistle  
whs



longwhistle  
whl



handclap  
hc



tambourine  
tamb



vibraslap  
vibs



tamtam  
tt



claves  
cl



hiwoodblock  
wbl







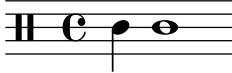
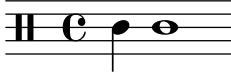








lowoodblock  
wbl



opencuica  
cuio





mutecuica cuim	triangle tri	opentriangle trio	mutetriangle trim
			
oneup ua	twoup ub	threeup uc	fourup ud
			
fiveup ue	onedown da	twodown db	threedown dc
			
fourdown dd	fivedown de		
			

## A.16 Glossario tecnico

Un glossario dei termini tecnici e dei concetti usati internamente in LilyPond. Questi termini appaiono nei manuali, nelle mailing list e nel codice sorgente.

### alist

Una lista di associazioni – **alist** per *association list* –, è una coppia Scheme che associa un valore a una parola chiave: (**chiave** . **valore**). Per esempio, in `scm/lily.scm`, la lista di associazioni “type-p-name-alist” associa alcuni tipi di predicato (come `ly:music?`) ai nomi (come “music”) in modo che gli errori relativi al controllo del tipo possano essere segnalati con un messaggio che includa il nome del tipo di predicato atteso.

### callback

**callback** indica una routine, funzione o metodo il cui riferimento sia passato come argomento quando si richiama un'altra routine, permettendo così alla routine richiamata di invocarla. La tecnica fa sì che a un livello più basso del software si possa richiamare una funzione definita a un livello più alto. I “callback” sono molto utilizzati in LilyPond per far sì che il codice Scheme a livello utente possa definire quante azioni di basso livello sono eseguite.

### closure

In Scheme, si parla di **closure** (chiusura) quando una funzione, di solito un'espressione lambda, viene passata come variabile. La chiusura contiene il codice della funzione più i riferimenti ai collegamenti lessicali delle variabili libere della funzione (ovvero quelle variabili usate nell'espressione ma definite al di fuori di essa). Quando questa funzione viene applicata a diversi argomenti successivamente, i collegamenti delle variabili libere che sono stati catturati nella

chiusura vengono usati per ottenere i valori delle variabili libere da usare nel calcolo. Una caratteristica utile delle chiusure è la conservazione dei valori delle variabili interne tra un’invocazione e l’altra, facendo sì che uno stato possa essere mantenuto.

## glyph

Un **glifo** è una particolare rappresentazione grafica di un carattere tipografico o una combinazione di due caratteri che formano una legatura. Un insieme di glifi con un solo stile e forma costituiscono un tipo di carattere (font), e un insieme di tipi di carattere con vari stili e dimensioni costituiscono una famiglia di caratteri tipografici.

## Vedi anche

Guida alla notazione: [\[Fonts\]](#), pagina [\[Special characters\]](#), pagina [\[Special characters\]](#).

## grob

Gli oggetti di LilyPond che rappresentano elementi della notazione nell’output – come teste di nota, gambi, legature di portamento e di valore, ditekking, chiavi, etc. – sono chiamati, in inglese, ‘Layout objects’ (‘Oggetti della formattazione’) o anche ‘GRaphical OBjects’ o **grobs** in forma breve. Sono rappresentati da istanze della classe **Grob**.

## Vedi anche

Manuale di apprendimento: Sezione “Oggetti e interfacce” in *Manuale di Apprendimento*, Sezione “Convenzioni per i nomi di oggetti e proprietà” in *Manuale di Apprendimento*, Sezione “Proprietà degli oggetti di formattazione” in *Manuale di Apprendimento*.

Guida al funzionamento interno: Sezione “grob-interface” in *Guida al Funzionamento Interno*, Sezione “All layout objects” in *Guida al Funzionamento Interno*.

## immutable

Un oggetto si dice **immutabile** – in inglese *immutable* – se il suo stato non può essere modificato dopo la sua creazione, in contrasto con un oggetto variabile, che può essere modificato dopo la sua creazione.

In LilyPond, le proprietà immutabili o condivise definiscono lo stile e il comportamento predefinito dei grob. Sono condivise tra molti oggetti. In apparente contraddizione col loro nome, possono essere modificate con `\override` e `\revert`.

## Vedi anche

Guida alla notazione: [\[mutable\]](#), pagina 746.

## interface

Le azioni e le proprietà comuni a un insieme di grob sono raggruppate in un oggetto chiamato **grob-interface** o semplicemente ‘interface’.

## Vedi anche

Manuale di apprendimento: Sezione “Oggetti e interfacce” in *Manuale di Apprendimento*, Sezione “Convenzioni per i nomi di oggetti e proprietà” in *Manuale di Apprendimento*, Sezione “Proprietà presenti nelle interfacce” in *Manuale di Apprendimento*.

Guida alla notazione: Sezione 5.2.2 [\[Layout interfaces\]](#), pagina 599.

Guida al funzionamento interno: Sezione “Graphical Object Interfaces” in *Guida al Funzionamento Interno*.

## lexer

A **lexer** is a program which converts a sequence of characters into a sequence of tokens, a process called lexical analysis. The LilyPond lexer converts the stream obtained from an input `.ly` file into a tokenized stream more suited to the next stage of processing - parsing, for which see [parser], pagina 746. The LilyPond lexer is built with Flex from the lexer file `lily/lexer.ll` which contains the lexical rules. This file is part of the source code and is not included in the LilyPond binary installation.

## mutable

Si dice che un oggetto è **variabile** – *mutable* in inglese – se il suo stato può essere modificato dopo la sua creazione, in contrasto con un oggetto immutabile, il cui stato viene fissato al momento della sua creazione.

In LilyPond, le proprietà variabili contengono valori specifici di un grob. Di solito, le liste di altri oggetti o i risultati di calcoli sono salvati in proprietà variabili.

## Vedi anche

Guida alla notazione: [immutable], pagina 745.

## output-def

Un'istanza della classe **Output-def** contiene i metodi e le strutture dei dati associate con un blocco di output. Tali istanze vengono create per i blocchi midi, layout e paper.

## parser

Un **analizzatore sintattico** – in inglese *parser* – analizza la sequenza di *token* prodotti da un *lexer* per determinare la sua struttura grammaticale, raggruppando i token progressivamente in gruppi più ampi in base a certe regole grammaticali. Se la sequenza di token è valida, il risultato finale è l'insieme dei token ordinati a albero, la cui radice è il simbolo iniziale della grammatica. Se ciò non può essere ottenuto, il file non è valido e viene generato un appropriato messaggio di errore. I gruppi sintattici e le regole che li definiscono nella sintassi di LilyPond sono definiti in `lily/parser.yy` e mostrati in Backus Normal Form (BNF) in Sezione “LilyPond grammar” in *Guida del Collaboratore*. Questo file viene usato dal generatore di parser Bison per generare il parser durante la compilazione del programma. Fa parte del codice sorgente e non è incluso nell'installazione binaria di LilyPond.

## parser variable

Si tratta di variabili definite direttamente in Scheme. Il loro uso da parte degli utente è fortemente scoraggiato, perché la semantica del loro raggio d'azione può creare confusione.

Se il valore di una simile variabile viene cambiato in un file `.ly`, la modifica è globale e, se non viene ripristinato esplicitamente, il nuovo valore viene mantenuto fino alla fine del file, agendo su blocchi `\score` successivi così come su file esterni aggiunti col comando `\include`. Ciò può portare a conseguenze non volute e in progetti complessi gli errori conseguenti possono essere difficili da individuare.

LilyPond usa le seguenti variabili dell'analizzatore sintattico:

- afterGraceFraction
- musicQuotes
- mode
- output-count
- output-suffix
- partCombineListener

- `pitchnames`
- `toplevel-bookparts`
- `toplevel-scores`
- `showLastLength`
- `showFirstLength`

## prob

Le proprietà dell'oggetto – **prob** per PProperty Objects – sono istanze della classe **Prob**, una semplice classe per oggetti che hanno liste associative di proprietà variabili e invariabili e metodi per manipolarle. Le classi **Music** e **Stream\_event** derivano da **Prob**. Vengono create istanze della classe **Prob** anche per conservare il contenuto formattato dei grob di un sistema e i blocchi dei titoli durante la formattazione della pagina.

## smob

Gli oggetti Scheme – **S mobs** per ScheMe OBjects – fanno parte del meccanismo con cui Guile esporta gli oggetti C e C++ in codice Scheme. In LilyPond, gli smob vengono creati dagli oggetti C++ attraverso delle macro. Esistono due tipi di oggetti smob: smob semplici, intesi per oggetti invariabili semplici come i numeri; e smob complessi, usati per oggetti aventi delle identità. Maggiori informazioni si trovano nei sorgenti di LilyPond e precisamente nel file `lily/includes/smob.hh`.

## stencil

Un'istanza della classe **stencil** contiene l'informazione necessaria per stampare un oggetto tipografico. È un semplice smob che contiene un riquadro che definisce l'estensione verticale e orizzontale dell'oggetto, e un'espressione Scheme che stamperà l'oggetto quando esaminata. Gli stencil possono essere combinati per formare stencil più complessi, definiti da una gerarchia di espressioni Scheme degli stencil che li compongono.

La proprietà `stencil`, che connette un grob al suo stencil, è definita nell'interfaccia `grob-interface`.

## Vedi anche

Guida al funzionamento interno: Sezione “grob-interface” in *Guida al Funzionamento Interno*.

## A.17 Tutte le proprietà di contesto

`accidentalGrouping` (symbol)

If set to 'voice, accidentals on the same note in different octaves may be horizontally staggered if in different voices.

`additionalPitchPrefix` (string)

Text with which to prefix additional pitches within a chord name.

`aDueText` (markup)

Text to print at a unisono passage.

`alignAboveContext` (string)

Where to insert newly created context in vertical alignment.

`alignBassFigureAccidentals` (boolean)

If true, then the accidentals are aligned in bass figure context.

`alignBelowContext` (string)

Where to insert newly created context in vertical alignment.

**alternativeNumberingStyle** (symbol)

The style of an alternative's bar numbers. Can be **numbers** for going back to the same number or **numbers-with-letters** for going back to the same number with letter suffixes. No setting will not go back in measure-number time.

**alternativeRestores** (symbol list)

Timing variables that are restored to their value at the end of the first alternative in subsequent alternatives.

**associatedVoice** (string)

Name of the context (see **associatedVoiceType** for its type, usually **Voice**) that has the melody for this **Lyrics** line.

**associatedVoiceType** (symbol)

Type of the context that has the melody for this **Lyrics** line.

**autoAccidentals** (list)

List of different ways to typeset an accidental.

For determining when to print an accidental, several different rules are tried. The rule that gives the highest number of accidentals is used.

Each entry in the list is either a symbol or a procedure.

*symbol* The symbol is the name of the context in which the following rules are to be applied. For example, if *context* is Sezione "Score" in *Guida al Funzionamento Interno* then all staves share accidentals, and if *context* is Sezione "Staff" in *Guida al Funzionamento Interno* then all voices in the same staff share accidentals, but staves do not.

*procedure* The procedure represents an accidental rule to be applied to the previously specified context.

The procedure takes the following arguments:

**context** The current context to which the rule should be applied.

**pitch** The pitch of the note to be evaluated.

**barnum** The current bar number.

**measurepos**

The current measure position.

The procedure returns a pair of booleans. The first states whether an extra natural should be added. The second states whether an accidental should be printed. (**#t** . **#f**) does not make sense.

**autoBeamCheck** (procedure)

A procedure taking three arguments, *context*, *dir* [start/stop (-1 or 1)], and *test* [shortest note in the beam]. A non-**#f** return value starts or stops the auto beam.

**autoBeaming** (boolean)

If set to true then beams are generated automatically.

**autoCautionaries** (list)

List similar to **autoAccidentals**, but it controls cautionary accidentals rather than normal ones. Both lists are tried, and the one giving the most accidentals wins. In case of draw, a normal accidental is typeset.

**automaticBars** (boolean)

If set to false then bar lines will not be printed automatically; they must be explicitly created with a **\bar** command. Unlike the **\cadenzaOn** keyword, measures are still

counted. Bar line generation will resume according to that count if this property is unset.

**barAlways** (boolean)

If set to true a bar line is drawn after each note.

**barCheckSynchronize** (boolean)

If true then reset **measurePosition** when finding a bar check.

**barNumberFormatter** (procedure)

A procedure that takes a bar number, measure position, and alternative number and returns a markup of the bar number to print.

**barNumberVisibility** (procedure)

A procedure that takes a bar number and a measure position and returns whether the corresponding bar number should be printed. Note that the actual print-out of bar numbers is controlled with the **break-visibility** property.

The following procedures are predefined:

**all-bar-numbers-visible**

Enable bar numbers for all bars, including the first one and broken bars (which get bar numbers in parentheses).

**first-bar-number-invisible**

Enable bar numbers for all bars (including broken bars) except the first one. If the first bar is broken, it doesn't get a bar number either.

**first-bar-number-invisible-save-broken-bars**

Enable bar numbers for all bars (including broken bars) except the first one. A broken first bar gets a bar number.

**first-bar-number-invisible-and-no-parenthesized-bar-numbers**

Enable bar numbers for all bars except the first bar and broken bars. This is the default.

**(every-nth-bar-number-visible *n*)**

Assuming *n* is value 2, for example, this enables bar numbers for bars 2, 4, 6, etc.

**(modulo-bar-number-visible *n m*)**

If bar numbers 1, 4, 7, etc., should be enabled, *n* (the modulo) must be set to 3 and *m* (the division remainder) to 1.

**baseMoment** (moment)

Smallest unit of time that will stand on its own as a subdivided section.

**bassFigureFormatFunction** (procedure)

A procedure that is called to produce the formatting for a **BassFigure** grob. It takes a list of **BassFigureEvents**, a context, and the grob to format.

**beamExceptions** (list)

An alist of exceptions to autobeam rules that normally end on beats.

**beamHalfMeasure** (boolean)

Whether to allow a beam to begin halfway through the measure in triple time, which could look like 6/8.

**beatStructure** (list)

List of **baseMoments** that are combined to make beats.

- chordChanges** (boolean)  
Only show changes in chords scheme?
- chordNameExceptions** (list)  
An alist of chord exceptions. Contains (*chord . markup*) entries.
- chordNameExceptionsFull** (list)  
An alist of full chord exceptions. Contains (*chord . markup*) entries.
- chordNameExceptionsPartial** (list)  
An alist of partial chord exceptions. Contains (*chord . (prefix-markup suffix-markup)*) entries.
- chordNameFunction** (procedure)  
The function that converts lists of pitches to chord names.
- chordNameLowercaseMinor** (boolean)  
Downcase roots of minor chords?
- chordNameSeparator** (markup)  
The markup object used to separate parts of a chord name.
- chordNoteNamer** (procedure)  
A function that converts from a pitch object to a text markup. Used for single pitches.
- chordPrefixSpacer** (number)  
The space added between the root symbol and the prefix of a chord name.
- chordRootNamer** (procedure)  
A function that converts from a pitch object to a text markup. Used for chords.
- clefGlyph** (string)  
Name of the symbol within the music font.
- clefPosition** (number)  
Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.
- clefTransposition** (integer)  
Add this much extra transposition. Values of 7 and -7 are common.
- clefTranspositionFormatter** (procedure)  
A procedure that takes the Transposition number as a string and the style as a symbol and returns a markup.
- clefTranspositionStyle** (symbol)  
Determines the way the ClefModifier grob is displayed. Possible values are ‘default’, ‘parenthesized’ and ‘bracketed’.
- completionBusy** (boolean)  
Whether a completion-note head is playing.
- completionFactor** (an exact rational or procedure)  
When `Completion_heads_engraver` and `Completion_rest_engraver` need to split a note or rest with a scaled duration, such as `c2*3`, this specifies the scale factor to use for the newly-split notes and rests created by the engraver.  
If `#f`, the completion engraver uses the scale-factor of each duration being split.  
If set to a callback procedure, that procedure is called with the context of the completion engraver, and the duration to be split.

- completionUnit** (moment)  
Sub-bar unit of completion.
- connectArpeggios** (boolean)  
If set, connect arpeggios across piano staff.
- countPercentRepeats** (boolean)  
If set, produce counters for percent repeats.
- createKeyOnClefChange** (boolean)  
Print a key signature whenever the clef is changed.
- createSpacing** (boolean)  
Create **StaffSpacing** objects? Should be set for staves.
- crescendoSpanner** (symbol)  
The type of spanner to be used for crescendi. Available values are ‘hairpin’ and ‘text’. If unset, a hairpin crescendo is used.
- crescendoText** (markup)  
The text to print at start of non-hairpin crescendo, i.e., ‘**cresc.**’.
- cueClefGlyph** (string)  
Name of the symbol within the music font.
- cueClefPosition** (number)  
Where should the center of the clef symbol go, measured in half staff spaces from the center of the staff.
- cueClefTransposition** (integer)  
Add this much extra transposition. Values of 7 and -7 are common.
- cueClefTranspositionFormatter** (procedure)  
A procedure that takes the Transposition number as a string and the style as a symbol and returns a markup.
- cueClefTranspositionStyle** (symbol)  
Determines the way the ClefModifier grob is displayed. Possible values are ‘default’, ‘parenthesized’ and ‘bracketed’.
- currentBarNumber** (integer)  
Contains the current barnumber. This property is incremented at every bar line.
- decrescendoSpanner** (symbol)  
The type of spanner to be used for decrescendi. Available values are ‘hairpin’ and ‘text’. If unset, a hairpin decrescendo is used.
- decrescendoText** (markup)  
The text to print at start of non-hairpin decrescendo, i.e., ‘**dim.**’.
- defaultBarType** (string)  
Set the default type of bar line. See **whichBar** for information on available bar types.  
This variable is read by Sezione “Timing.translator” in *Guida al Funzionamento Interno* at Sezione “Score” in *Guida al Funzionamento Interno* level.
- defaultStrings** (list)  
A list of strings to use in calculating frets for tablatures and fretboards if no strings are provided in the notes for the current moment.
- doubleRepeatSegnoType** (string)  
Set the default bar line for the combinations double repeat with segno. Default is ‘:|.S.|:’.



**doubleRepeatType** (string)

Set the default bar line for double repeats.

**doubleSlurs** (boolean)

If set, two slurs are created for every slurred note, one above and one below the chord.

**drumPitchTable** (hash table)

A table mapping percussion instruments (symbols) to pitches.

**drumStyleTable** (hash table)

A hash table which maps drums to layout settings. Predefined values: ‘drums-style’, ‘agostini-drums-style’, ‘timbales-style’, ‘congas-style’, ‘bongos-style’, and ‘percussion-style’.

The layout style is a hash table, containing the drum-pitches (e.g., the symbol ‘hihat’) as keys, and a list (*notehead-style script vertical-position*) as values.

**endRepeatSegnoType** (string)

Set the default bar line for the combinations ending of repeat with segno. Default is ‘:|.S’.

**endRepeatType** (string)

Set the default bar line for the ending of repeats.

**explicitClefVisibility** (vector)

‘break-visibility’ function for clef changes.

**explicitCueClefVisibility** (vector)

‘break-visibility’ function for cue clef changes.

**explicitKeySignatureVisibility** (vector)

‘break-visibility’ function for explicit key changes. ‘\override’ of the *break-visibility* property will set the visibility for normal (i.e., at the start of the line) key signatures.

**extendersOverRests** (boolean)

Whether to continue extenders as they cross a rest.

**extraNatural** (boolean)

Whether to typeset an extra natural sign before accidentals that reduce the effect of a previous alteration.

**figuredBassAlterationDirection** (direction)

Where to put alterations relative to the main figure.

**figuredBassCenterContinuations** (boolean)

Whether to vertically center pairs of extender lines. This does not work with three or more lines.

**figuredBassFormatter** (procedure)

A routine generating a markup for a bass figure.

**figuredBassPlusDirection** (direction)

Where to put plus signs relative to the main figure.

**fingeringOrientations** (list)

A list of symbols, containing ‘left’, ‘right’, ‘up’ and/or ‘down’. This list determines where fingerings are put relative to the chord being fingered.

- firstClef** (boolean)  
If true, create a new clef when starting a staff.
- followVoice** (boolean)  
If set, note heads are tracked across staff switches by a thin line.
- fontSize** (number)  
The relative size of all grobs in a context.
- forbidBreak** (boolean)  
If set to **#t**, prevent a line break at this point.
- forceClef** (boolean)  
Show clef symbol, even if it has not changed. Only active for the first clef after the property is set, not for the full staff.
- fretLabels** (list)  
A list of strings or Scheme-formatted markups containing, in the correct order, the labels to be used for lettered frets in tablature.
- glissandoMap** (list)  
A map in the form of '((source1 . target1) (source2 . target2) (sourcen . targetn)) showing the glissandi to be drawn for note columns. The value '()' will default to '((0 . 0) (1 . 1) (n . n)), where n is the minimal number of note-heads in the two note columns between which the glissandi occur.
- gridInterval** (moment)  
Interval for which to generate **GridPoints**.
- handleNegativeFrets** (symbol)  
How the automatic fret calculator should handle calculated negative frets. Values include **'ignore**, to leave them out of the diagram completely, **'include**, to include them as calculated, and **'recalculate**, to ignore the specified string and find a string where they will fit with a positive fret number.
- harmonicAccidentals** (boolean)  
If set, harmonic notes in chords get accidentals.
- harmonicDots** (boolean)  
If set, harmonic notes in dotted chords get dots.
- highStringOne** (boolean)  
Whether the first string is the string with highest pitch on the instrument. This used by the automatic string selector for tablature notation.
- ignoreBarChecks** (boolean)  
Ignore bar checks.
- ignoreFiguredBassRest** (boolean)  
Don't swallow rest events.
- ignoreMelismata** (boolean)  
Ignore melismata for this Sezione "Lyrics" in *Guida al Funzionamento Interno* line.
- implicitBassFigures** (list)  
A list of bass figures that are not printed as numbers, but only as extender lines.
- includeGraceNotes** (boolean)  
Do not ignore grace notes for Sezione "Lyrics" in *Guida al Funzionamento Interno*.
- initialTimeSignatureVisibility** (vector)  
break visibility for the initial time signature.

**instrumentCueName** (markup)

The name to print if another instrument is to be taken.

**instrumentEqualizer** (procedure)

A function taking a string (instrument name), and returning a (*min* . *max*) pair of numbers for the loudness range of the instrument.

**instrumentName** (markup)

The name to print left of a staff. The **instrumentName** property labels the staff in the first system, and the **shortInstrumentName** property labels following lines.

**instrumentTransposition** (pitch)

Define the transposition of the instrument. Its value is the pitch that sounds when the instrument plays written middle C. This is used to transpose the MIDI output, and \quotes.

**internalBarNumber** (integer)

Contains the current barnumber. This property is used for internal timekeeping, among others by the **Accidental\_engraver**.

**keepAliveInterfaces** (list)

A list of symbols, signifying grob interfaces that are worth keeping a staff with **remove-empty** set around for.

**keyAlterationOrder** (list)

An alist that defines in what order alterations should be printed. The format is (*step* . *alter*), where *step* is a number from 0 to 6 and *alter* from -2 (sharp) to 2 (flat).

**keyAlterations** (list)

The current key signature. This is an alist containing (*step* . *alter*) or ((*octave* . *step*) . *alter*), where *step* is a number in the range 0 to 6 and *alter* a fraction, denoting alteration. For alterations, use symbols, e.g. **keyAlterations** = #`((6 . ,FLAT)).

**lyricMelismaAlignment** (number)

Alignment to use for a melisma syllable.

**magnifyStaffValue** (positive number)

The most recent value set with **\magnifyStaff**.

**majorSevenSymbol** (markup)

How should the major 7th be formatted in a chord name?

**markFormatter** (procedure)

A procedure taking as arguments the context and the rehearsal mark. It should return the formatted mark as a markup object.

**maximumFretStretch** (number)

Don't allocate frets further than this from specified frets.

**measureLength** (moment)

Length of one measure in the current time signature.

**measurePosition** (moment)

How much of the current measure have we had. This can be set manually to create incomplete measures.

**melismaBusyProperties** (list)

A list of properties (symbols) to determine whether a melisma is playing. Setting this property will influence how lyrics are aligned to notes. For example,

if set to '(melismaBusy beamMelismaBusy), only manual melismata and manual beams are considered. Possible values include `melismaBusy`, `slurMelismaBusy`, `tieMelismaBusy`, and `beamMelismaBusy`.

`metronomeMarkFormatter` (procedure)

How to produce a metronome markup. Called with two arguments: a `TempoChangeEvent` and context.

`middleCClefPosition` (number)

The position of the middle C, as determined only by the clef. This can be calculated by looking at `clefPosition` and `clefGlyph`.

`middleCCuePosition` (number)

The position of the middle C, as determined only by the clef of the cue notes. This can be calculated by looking at `cueClefPosition` and `cueClefGlyph`.

`middleCOffset` (number)

The offset of middle C from the position given by `middleCClefPosition`. This is used for ottava brackets.

`middleCPosition` (number)

The place of the middle C, measured in half staff-spaces. Usually determined by looking at `middleCClefPosition` and `middleCOffset`.

`midiBalance` (number)

Stereo balance for the MIDI channel associated with the current context. Ranges from -1 to 1, where the values -1 (`#LEFT`), 0 (`#CENTER`) and 1 (`#RIGHT`) correspond to leftmost emphasis, center balance, and rightmost emphasis, respectively.

`midiChannelMapping` (symbol)

How to map MIDI channels: per `staff` (default), `instrument` or `voice`.

`midiChorusLevel` (number)

Chorus effect level for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`midiExpression` (number)

Expression control for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`midiInstrument` (string)

Name of the MIDI instrument to use.

`midiMaximumVolume` (number)

Analogous to `midiMinimumVolume`.

`midiMergeUnisons` (boolean)

If true, output only one MIDI note-on event when notes with the same pitch, in the same MIDI-file track, overlap.

`midiMinimumVolume` (number)

Set the minimum loudness for MIDI. Ranges from 0 to 1.

`midiPanPosition` (number)

Pan position for the MIDI channel associated with the current context. Ranges from -1 to 1, where the values -1 (`#LEFT`), 0 (`#CENTER`) and 1 (`#RIGHT`) correspond to hard left, center, and hard right, respectively.

`midiReverbLevel` (number)

Reverb effect level for the MIDI channel associated with the current context. Ranges from 0 to 1 (0=off, 1=full effect).

`minimumFret` (number)

The tablature auto string-selecting mechanism selects the highest string with a fret at least `minimumFret`.

`minimumPageTurnLength` (moment)

Minimum length of a rest for a page turn to be allowed.

`minimumRepeatLengthForPageTurn` (moment)

Minimum length of a repeated section for a page turn to be allowed within that section.

`minorChordModifier` (markup)

Markup displayed following the root for a minor chord

`noChordSymbol` (markup)

Markup to be displayed for rests in a `ChordNames` context.

`noteToFretFunction` (procedure)

Convert list of notes and list of defined strings to full list of strings and fret numbers. Parameters: The context, a list of note events, a list of tabstring events, and the fretboard grob if a fretboard is desired.

`nullAccidentals` (boolean)

The `Accidental_engraver` generates no accidentals for notes in contexts where this is set. In addition to suppressing the printed accidental, this option removes any effect the note would have had on accidentals in other voices.

`ottavation` (markup)

If set, the text for an ottava spanner. Changing this creates a new text spanner.

`output` (music output)

The output produced by a score-level translator during music interpretation.

`partCombineForced` (symbol)

Override for the `partcombine` decision. Can be `apart`, `chords`, `unisono`, `solo1`, or `solo2`.

`partCombineTextsOnNote` (boolean)

Print part-combine texts only on the next note rather than immediately on rests or skips.

`pedalSostenutoStrings` (list)

See `pedalSustainStrings`.

`pedalSostenutoStyle` (symbol)

See `pedalSustainStyle`.

`pedalSustainStrings` (list)

A list of strings to print for sustain-pedal. Format is *(up updown down)*, where each of the three is the string to print when this is done with the pedal.

`pedalSustainStyle` (symbol)

A symbol that indicates how to print sustain pedals: `text`, `bracket` or `mixed` (both).

`pedalUnaCordaStrings` (list)

See `pedalSustainStrings`.

`pedalUnaCordaStyle` (symbol)

See `pedalSustainStyle`.

`predefinedDiagramTable` (hash table)

The hash table of predefined fret diagrams to use in `FretBoards`.

**printKeyCancellation** (boolean)  
Print restoration alterations before a key signature change.

**printOctaveNames** (boolean)  
Print octave marks for the **NoteNames** context.

**printPartCombineTexts** (boolean)  
Set ‘Solo’ and ‘A due’ texts in the part combiner?

**proportionalNotationDuration** (moment)  
Global override for shortest-playing duration. This is used for switching on proportional notation.

**rehearsalMark** (integer)  
The last rehearsal mark printed.

**repeatCommands** (list)  
This property is a list of commands of the form (list 'volta x), where x is a string or #f. 'end-repeat is also accepted as a command.

**repeatCountVisibility** (procedure)  
A procedure taking as arguments an integer and context, returning whether the corresponding percent repeat number should be printed when **countPercentRepeats** is set.

**restCompletionBusy** (boolean)  
Signal whether a completion-rest is active.

**restNumberThreshold** (number)  
If a multimeasure rest has more measures than this, a number is printed.

**restrainOpenStrings** (boolean)  
Exclude open strings from the automatic fret calculator.

**searchForVoice** (boolean)  
Signal whether a search should be made of all contexts in the context hierarchy for a voice to provide rhythms for the lyrics.

**segnoType** (string)  
Set the default bar line for a requested segno. Default is ‘S’.

**shapeNoteStyles** (vector)  
Vector of symbols, listing style for each note head relative to the tonic (qv.) of the scale.

**shortInstrumentName** (markup)  
See **instrumentName**.

**shortVocalName** (markup)  
Name of a vocal line, short version.

**skipBars** (boolean)  
If set to true, then skip the empty bars that are produced by multimeasure notes and rests. These bars will not appear on the printed output. If not set (the default), multimeasure notes and rests expand into their full length, printing the appropriate number of empty bars so that synchronization with other voices is preserved.

```
{
  r1 r1*3 R1*3
  \set Score.skipBars= ##t
  r1*3 R1*3
}
```

**skipTypesetting** (boolean)

If true, no typesetting is done, speeding up the interpretation phase. Useful for debugging large scores.

**slashChordSeparator** (markup)

The markup object used to separate a chord name from its root note in case of inversions or slash chords.

**soloIIIText** (markup)

The text for the start of a solo for voice ‘two’ when part-combining.

**soloText** (markup)

The text for the start of a solo when part-combining.

**squashedPosition** (integer)

Vertical position of squashing for Sezione “Pitch\_squash\_engraver” in *Guida al Funzionamento Interno*.

**staffLineLayoutFunction** (procedure)

Layout of staff lines, **traditional**, or **semitone**.

**stanza** (markup)

Stanza ‘number’ to print before the start of a verse. Use in **Lyrics** context.

**startRepeatSegnoType** (string)

Set the default bar line for the combinations beginning of repeat with segno. Default is ‘S.|:’.

**startRepeatType** (string)

Set the default bar line for the beginning of repeats.

**stemLeftBeamCount** (integer)

Specify the number of beams to draw on the left side of the next note. Overrides automatic beaming. The value is only used once, and then it is erased.

**stemRightBeamCount** (integer)

See **stemLeftBeamCount**.

**strictBeatBeaming** (boolean)

Should partial beams reflect the beat structure even if it causes flags to hang out?

**stringNumberOrientations** (list)

See **fingeringOrientations**.

**stringOneTopmost** (boolean)

Whether the first string is printed on the top line of the tablature.

**stringTunings** (list)

The tablature strings tuning. It is a list of the pitches of each string (starting with the lowest numbered one).

**strokeFingerOrientations** (list)

See **fingeringOrientations**.

**subdivideBeams** (boolean)

If set, multiple beams will be subdivided at **baseMoment** positions by only drawing one beam over the beat.

**suggestAccidentals** (boolean)

If set, accidentals are typeset as cautionary suggestions over the note.

**supportNonIntegerFret** (boolean)

If set in **Score** the **TabStaff** will print micro-tones as ‘2 $\frac{1}{2}$ ’

**systemStartDelimiter** (symbol)

Which grob to make for the start of the system/staff? Set to **SystemStartBrace**, **SystemStartBracket** or **SystemStartBar**.

**systemStartDelimiterHierarchy** (pair)

A nested list, indicating the nesting of a start delimiters.

**tablatureFormat** (procedure)

A function formatting a tablature note head. Called with three arguments: context, string number and, fret number. It returns the text as a markup.

**tabStaffLineLayoutFunction** (procedure)

A function determining the staff position of a tablature note head. Called with two arguments: the context and the string.

**tempoHideNote** (boolean)

Hide the note = count in tempo marks.

**tempoWholesPerMinute** (moment)

The tempo in whole notes per minute.

**tieWaitForNote** (boolean)

If true, tied notes do not have to follow each other directly. This can be used for writing out arpeggios.

**timeSignatureFraction** (fraction, as pair)

A pair of numbers, signifying the time signature. For example, '(4 . 4) is a 4/4 time signature.

**timeSignatureSettings** (list)

A nested alist of settings for time signatures. Contains elements for various time signatures. The element for each time signature contains entries for **baseMoment**, **beatStructure**, and **beamExceptions**.

**timing** (boolean)

Keep administration of measure length, position, bar number, etc.? Switch off for cadenzas.

**tonic** (pitch)

The tonic of the current scale.

**topLevelAlignment** (boolean)

If true, the *Vertical-align-engraver* will create a *VerticalAlignment*; otherwise, it will create a *StaffGrouper*

**tupletFullLength** (boolean)

If set, the tuplet is printed up to the start of the next note.

**tupletFullLengthNote** (boolean)

If set, end at the next note, otherwise end on the matter (time signatures, etc.) before the note.

**tupletSpannerDuration** (moment)

Normally, a tuplet bracket is as wide as the `\times` expression that gave rise to it. By setting this property, you can make brackets last shorter.

```
{
  \set tupletSpannerDuration = #(ly:make-moment 1 4)
  \times 2/3 { c8 c c c c c }
}
```



`useBassFigureExtenders` (boolean)

Whether to use extender lines for repeated bass figures.

`vocalName` (markup)

Name of a vocal line.

`voltaSpannerDuration` (moment)

This specifies the maximum duration to use for the brackets printed for `\alternative`. This can be used to shrink the length of brackets in the situation where one alternative is very large.

`whichBar` (string)

This property is read to determine what type of bar line to create.

Example:

```
\set Staff.whichBar = ".|:"
```

This will create a start-repeat bar in this staff only. Valid values are described in `scm/bar-line.scm`.

## A.18 Proprietà della formattazione

`add-stem-support` (boolean)

If set, the `Stem` object is included in this script's support.

`after-line-breaking` (boolean)

Dummy property, used to trigger callback for `after-line-breaking`.

`align-dir` (direction)

Which side to align? -1: left side, 0: around center of width, 1: right side.

`allow-loose-spacing` (boolean)

If set, column can be detached from main spacing.

`allow-span-bar` (boolean)

If false, no inter-staff bar line will be created below this bar line.

`alteration` (number)

Alteration numbers for accidental.

`alteration-alist` (list)

List of (*pitch* . *accidental*) pairs for key signature.

`annotation` (string)

Annotate a grob for debug purposes.

`annotation-balloon` (boolean)

Print the balloon around an annotation.

`annotation-line` (boolean)

Print the line from an annotation to the grob that it annotates.

`arpeggio-direction` (direction)

If set, put an arrow on the arpeggio squiggly line.

`arrow-length` (number)

Arrow length.

`arrow-width` (number)

Arrow width.

**auto-knee-gap** (dimension, in staff space)

If a gap is found between note heads where a horizontal beam fits that is larger than this number, make a kneed beam.

**automatically-numbered** (boolean)

Should a footnote be automatically numbered?

**average-spacing-wishes** (boolean)

If set, the spacing wishes are averaged over staves.

**avoid-note-head** (boolean)

If set, the stem of a chord does not pass through all note heads, but starts at the last note head.

**avoid-scripts** (boolean)

If set, a tuplet bracket avoids the scripts associated with the note heads it encompasses.

**avoid-slur** (symbol)

Method of handling slur collisions. Choices are **inside**, **outside**, **around**, and **ignore**. **inside** adjusts the slur if needed to keep the grob inside the slur. **outside** moves the grob vertically to the outside of the slur. **around** moves the grob vertically to the outside of the slur only if there is a collision. **ignore** does not move either. In grobs whose notational significance depends on vertical position (such as accidentals, clefs, etc.), **outside** and **around** behave like **ignore**.

**axes** (list) List of axis numbers. In the case of alignment grobs, this should contain only one number.

**bar-extent** (pair of numbers)

The Y-extent of the actual bar line. This may differ from **Y-extent** because it does not include the dots in a repeat bar line.

**base-shortest-duration** (moment)

Spacing is based on the shortest notes in a piece. Normally, pieces are spaced as if notes at least as short as this are present.

**baseline-skip** (dimension, in staff space)

Distance between base lines of multiple lines of text.

**beam-thickness** (dimension, in staff space)

Beam thickness, measured in **staff-space** units.

**beam-width** (dimension, in staff space)

Width of the tremolo sign.

**beamed-stem-shorten** (list)

How much to shorten beamed stems, when their direction is forced. It is a list, since the value is different depending on the number of flags and beams.

**beaming** (pair)

Pair of number lists. Each number list specifies which beams to make. 0 is the central beam, 1 is the next beam toward the note, etc. This information is used to determine how to connect the beaming patterns from stem to stem inside a beam.

**beamlet-default-length** (pair)

A pair of numbers. The first number specifies the default length of a beamlet that sticks out of the left hand side of this stem; the second number specifies the default length of the beamlet to the right. The actual length of a beamlet is determined by taking either the default length or the length specified by **beamlet-max-length-proportion**, whichever is smaller.

**beamlet-max-length-proportion** (pair)

The maximum length of a beamlet, as a proportion of the distance between two adjacent stems.

**before-line-breaking** (boolean)

Dummy property, used to trigger a callback function.

**between-cols** (pair)

Where to attach a loose column to.

**bound-details** (list)

An alist of properties for determining attachments of spanners to edges.

**bound-padding** (number)

The amount of padding to insert around spanner bounds.

**bracket-flare** (pair of numbers)

A pair of numbers specifying how much edges of brackets should slant outward. Value 0.0 means straight edges.

**bracket-visibility** (boolean or symbol)

This controls the visibility of the tuplet bracket. Setting it to false prevents printing of the bracket. Setting the property to `if-no-beam` makes it print only if there is no beam associated with this tuplet bracket.

**break-align-anchor** (number)

Grobs aligned to this breakable item will have their X-offsets shifted by this number. In bar lines, for example, this is used to position grobs relative to the (visual) center of the bar line.

**break-align-anchor-alignment** (number)

Read by `ly:break-aligned-interface::calc-extent-aligned-anchor` for aligning an anchor to a grob's extent.

**break-align-orders** (vector)

This is a vector of 3 lists:  `#(end-of-line unbroken start-of-line)`. Each list contains *break-align symbols* that specify an order of breakable items (see Sezione “break-alignment-interface” in *Guida al Funzionamento Interno*).

For example, this places time signatures before clefs:

```
\override Score.BreakAlignment.break-align-orders =
  #(make-vector 3 '(left-edge
                    cue-end-clef
                    ambitus
                    breathing-sign
                    time-signature
                    clef
                    cue-clef
                    staff-bar
                    key-cancellation
                    key-signature
                    custos))
```

**break-align-symbol** (symbol)

This key is used for aligning, ordering, and spacing breakable items. See Sezione “break-alignment-interface” in *Guida al Funzionamento Interno*.

**break-align-symbols** (list)

A list of *break-align symbols* that determines which breakable items to align this to. If the grob selected by the first symbol in the list is invisible due to **break-visibility**,

we will align to the next grob (and so on). Choices are listed in Sezione “break-alignment-interface” in *Guida al Funzionamento Interno*.

**break-overshoot** (pair of numbers)

How much does a broken spanner stick out of its bounds?

**break-visibility** (vector)

A vector of 3 booleans, **#(end-of-line unbroken begin-of-line)**. **#t** means visible, **#f** means killed.

**breakable** (boolean)

Allow breaks here.

**broken-bound-padding** (number)

The amount of padding to insert when a spanner is broken at a line break.

**chord-dots-limit** (integer)

Limits the column of dots on each chord to the height of the chord plus **chord-dots-limit** staff-positions.

**circled-tip** (boolean)

Put a circle at start/end of hairpins (al/del niente).

**clef-alignments** (list)

An alist of parent-alignments that should be used for clef modifiers with various clefs

**clip-edges** (boolean)

Allow outward pointing beamlets at the edges of beams?

**collapse-height** (dimension, in staff space)

Minimum height of system start delimiter. If equal or smaller, the bracket/brace/line is removed.

**collision-interfaces** (list)

A list of interfaces for which automatic beam-collision resolution is run.

**collision-voice-only** (boolean)

Does automatic beam collision apply only to the voice in which the beam was created?

**color** (color)

The color of this grob.

**common-shortest-duration** (moment)

The most common shortest note length. This is used in spacing. Enlarging this sets the score tighter.

**concaveness** (number)

A beam is concave if its inner stems are closer to the beam than the two outside stems. This number is a measure of the closeness of the inner stems. It is used for damping the slope of the beam.

**connect-to-neighbor** (pair)

Pair of booleans, indicating whether this grob looks as a continued break.

**control-points** (list of number pairs)

List of offsets (number pairs) that form control points for the tie, slur, or bracket shape. For Béziers, this should list the control points of a third-order Bézier curve.

**count-from** (integer)

The first measure in a measure count receives this number. The following measures are numbered in increments from this initial value.

**damping** (number)

Amount of beam slope damping.

**dash-definition** (pair)

List of **dash-elements** defining the dash structure. Each **dash-element** has a starting t value, an ending t-value, a **dash-fraction**, and a **dash-period**.

**dash-fraction** (number)

Size of the dashes, relative to **dash-period**. Should be between 0.1 and 1.0 (continuous line). If set to 0.0, a dotted line is produced

**dash-period** (number)

The length of one dash together with whitespace. If negative, no line is drawn at all.

**default-direction** (direction)

Direction determined by note head positions.

**default-staff-staff-spacing** (list)

The settings to use for **staff-staff-spacing** when it is unset, for ungrouped staves and for grouped staves that do not have the relevant **StaffGrouper** property set (**staff-staff-spacing** or **staffgroup-staff-spacing**).

**details** (list)

Alist of parameters for detailed grob behavior. More information on the allowed parameters for a grob can be found by looking at the top of the Internals Reference page for each interface having a **details** property.

**digit-names** (vector)

Names for string finger digits.

**direction** (direction)

If **side-axis** is 0 (or X), then this property determines whether the object is placed LEFT, CENTER or RIGHT with respect to the other object. Otherwise, it determines whether the object is placed UP, CENTER or DOWN. Numerical values may also be used: UP=1, DOWN=-1, LEFT=-1, RIGHT=1, CENTER=0.

**dot-count** (integer)

The number of dots.

**dot-negative-kern** (number)

The space to remove between a dot and a slash in percent repeat glyphs. Larger values bring the two elements closer together.

**dot-placement-list** (list)

List consisting of (*description string-number fret-number finger-number*) entries used to define fret diagrams.

**double-stem-separation** (number)

The distance between the two stems of a half note in tablature when using `\tabFullNotation`, not counting the width of the stems themselves, expressed as a multiple of the default height of a staff-space in the traditional five-line staff.

**duration-log** (integer)

The 2-log of the note head duration, i.e., 0 = whole note, 1 = half note, etc.

**eccentricity** (number)

How asymmetrical to make a slur. Positive means move the center to the right.

**edge-height** (pair)

A pair of numbers specifying the heights of the vertical edges: (*left-height* . *right-height*).

**edge-text** (pair)

A pair specifying the texts to be set at the edges: (*left-text* . *right-text*).

**expand-limit** (integer)

Maximum number of measures expanded in church rests.

**extra-dy** (number)

Slope glissandi this much extra.

**extra-offset** (pair of numbers)

A pair representing an offset. This offset is added just before outputting the symbol, so the typesetting engine is completely oblivious to it. The values are measured in **staff-space** units of the staff's **StaffSymbol**.

**extra-spacing-height** (pair of numbers)

In the horizontal spacing problem, we increase the height of each item by this amount (by adding the 'car' to the bottom of the item and adding the 'cdr' to the top of the item). In order to make a grob infinitely high (to prevent the horizontal spacing problem from placing any other grobs above or below this grob), set this to (*-inf.0* . *+inf.0*).

**extra-spacing-width** (pair of numbers)

In the horizontal spacing problem, we pad each item by this amount (by adding the 'car' on the left side of the item and adding the 'cdr' on the right side of the item). In order to make a grob take up no horizontal space at all, set this to (*+inf.0* . *-inf.0*).

**flag-count** (number)

The number of tremolo beams.

**flag-style** (symbol)

The style of the flag to be used with **MetronomeMark**. Available are 'modern-straight-flag', 'old-straight-flag', flat-flag, mensural and 'default'

**flat-positions** (list)

Flats in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: (**alto** **treble** **tenor** **soprano** **baritone** **mezzosoprano** **bass**). If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

**font-encoding** (symbol)

The font encoding is the broadest category for selecting a font. Currently, only Lilypond's system fonts (Emmentaler) are using this property. Available values are **fetaMusic** (Emmentaler), **fetaBraces**, **fetaText** (Emmentaler).

**font-family** (symbol)

The font family is the broadest category for selecting text fonts. Options include: **sans**, **roman**.

**font-name** (string)

Specifies a file name (without extension) of the font to load. This setting overrides selection using **font-family**, **font-series** and **font-shape**.

**font-series** (symbol)

Select the series of a font. Choices include **medium**, **bold**, **bold-narrow**, etc.

**font-shape** (symbol)

Select the shape of a font. Choices include **upright**, **italic**, **caps**.

**font-size** (number)

The font size, compared to the ‘normal’ size. 0 is style-sheet’s normal size, -1 is smaller, +1 is bigger. Each step of 1 is approximately 12% larger; 6 steps are exactly a factor 2 larger. If the context property **fontSize** is set, its value is added to this before the glyph is printed. Fractional values are allowed.

**footnote** (boolean)

Should this be a footnote or in-note?

**footnote-music** (music)

Music creating a footnote.

**footnote-text** (markup)

A footnote for the grob.

**force-hshift** (number)

This specifies a manual shift for notes in collisions. The unit is the note head width of the first voice note. This is used by Sezione “note-collision-interface” in *Guida al Funzionamento Interno*.

**forced-spacing** (number)

Spacing forced between grobs, used in various ligature engravers.

**fraction** (fraction, as pair)

Numerator and denominator of a time signature object.

**french-beaming** (boolean)

Use French beaming style for this stem. The stem stops at the innermost beams.

**fret-diagram-details** (list)

An alist of detailed grob properties for fret diagrams. Each alist entry consists of a (**property** . **value**) pair. The properties which can be included in **fret-diagram-details** include the following:

- **barre-type** – Type of barre indication used. Choices include **curved**, **straight**, and **none**. Default **curved**.
- **capo-thickness** – Thickness of capo indicator, in multiples of fret-space. Default value 0.5.
- **dot-color** – Color of dots. Options include **black** and **white**. Default **black**.
- **dot-label-font-mag** – Magnification for font used to label fret dots. Default value 1.
- **dot-position** – Location of dot in fret space. Default 0.6 for dots without labels, 0.95-**dot-radius** for dots with labels.
- **dot-radius** – Radius of dots, in terms of fret spaces. Default value 0.425 for labeled dots, 0.25 for unlabeled dots.
- **finger-code** – Code for the type of fingering indication used. Options include **none**, **in-dot**, and **below-string**. Default **none** for markup fret diagrams, **below-string** for **FretBoards** fret diagrams.
- **fret-count** – The number of frets. Default 4.
- **fret-label-custom-format** – The format string to be used label the lowest fret number, when **number-type** equals to **custom**. Default “~a”.

- **fret-label-font-mag** – The magnification of the font used to label the lowest fret number. Default 0.5.
- **fret-label-vertical-offset** – The offset of the fret label from the center of the fret in direction parallel to strings. Default 0.
- **fret-label-horizontal-offset** – The offset of the fret label from the center of the fret in direction orthogonal to strings. Default 0.
- **paren-padding** – The padding for the parenthesis. Default 0.05.
- **label-dir** – Side to which the fret label is attached. -1, LEFT, or DOWN for left or down; 1, RIGHT, or UP for right or up. Default RIGHT.
- **mute-string** – Character string to be used to indicate muted string. Default "x".
- **number-type** – Type of numbers to use in fret label. Choices include **roman-lower**, **roman-upper**, **arabic** and **custom**. In the later case, the format string is supplied by the **fret-label-custom-format** property. Default **roman-lower**.
- **open-string** – Character string to be used to indicate open string. Default "o".
- **orientation** – Orientation of fret-diagram. Options include **normal**, **landscape**, and **opposing-landscape**. Default **normal**.
- **string-count** – The number of strings. Default 6.
- **string-label-font-mag** – The magnification of the font used to label fingerings at the string, rather than in the dot. Default value 0.6 for **normal** orientation, 0.5 for **landscape** and **opposing-landscape**.
- **string-thickness-factor** – Factor for changing thickness of each string in the fret diagram. Thickness of string  $k$  is given by **thickness** \*  $(1 + \text{string-thickness-factor})^{(k-1)}$ . Default 0.
- **top-fret-thickness** – The thickness of the top fret line, as a multiple of the standard thickness. Default value 3.
- **xo-font-magnification** – Magnification used for mute and open string indicators. Default value 0.5.
- **xo-padding** – Padding for open and mute indicators from top fret. Default value 0.25.

**full-length-padding** (number)

How much padding to use at the right side of a full-length tuplet bracket.

**full-length-to-extent** (boolean)

Run to the extent of the column for a full-length tuplet bracket.

**full-measure-extra-space** (number)

Extra space that is allocated at the beginning of a measure with only one note. This property is read from the **NonMusicalPaperColumn** that begins the measure.

**full-size-change** (boolean)

Don't make a change clef smaller.

**gap** (dimension, in staff space)

Size of a gap in a variable symbol.

**gap-count** (integer)

Number of gapped beams for tremolo.



**glissando-skip** (boolean)

Should this **NoteHead** be skipped by glissandi?

**glyph** (string)

A string determining what ‘style’ of glyph is typeset. Valid choices depend on the function that is reading this property.

In combination with (span) bar lines, it is a string resembling the bar line appearance in ASCII form.

**glyph-name** (string)

The glyph name within the font.

In the context of (span) bar lines, *glyph-name* represents a processed form of *glyph*, where decisions about line breaking etc. are already taken.

**glyph-name-alist** (list)

An alist of key-string pairs.

**graphical** (boolean)

Display in graphical (vs. text) form.

**grow-direction** (direction)

Crescendo or decrescendo?

**hair-thickness** (number)

Thickness of the thin line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to *Staff.StaffSymbol.thickness*).

**harp-pedal-details** (list)

An alist of detailed grob properties for harp pedal diagrams. Each alist entry consists of a (*property* . *value*) pair. The properties which can be included in harp-pedal-details include the following:

- **box-offset** – Vertical shift of the center of flat/sharp pedal boxes above/below the horizontal line. Default value 0.8.
- **box-width** – Width of each pedal box. Default value 0.4.
- **box-height** – Height of each pedal box. Default value 1.0.
- **space-before-divider** – Space between boxes before the first divider (so that the diagram can be made symmetric). Default value 0.8.
- **space-after-divider** – Space between boxes after the first divider. Default value 0.8.
- **circle-thickness** – Thickness (in unit of the line-thickness) of the ellipse around circled pedals. Default value 0.5.
- **circle-x-padding** – Padding in X direction of the ellipse around circled pedals. Default value 0.15.
- **circle-y-padding** – Padding in Y direction of the ellipse around circled pedals. Default value 0.2.

**head-direction** (direction)

Are the note heads left or right in a semitie?

**height** (dimension, in staff space)

Height of an object in **staff-space** units.

**height-limit** (dimension, in staff space)

Maximum slur height: The longer the slur, the closer it is to this height.

**hide-tied-accidental-after-break** (boolean)

If set, an accidental that appears on a tied note after a line break will not be displayed.

**horizon-padding** (number)

The amount to pad the axis along which a Skyline is built for the **side-position-interface**.

**horizontal-shift** (integer)

An integer that identifies ranking of **NoteColumns** for horizontal shifting. This is used by Sezione “note-collision-interface” in *Guida al Funzionamento Interno*.

**horizontal-skylines** (pair of skylines)

Two skylines, one to the left and one to the right of this grob.

**id** (string)

An id string for the grob. Depending on the typesetting backend being used, this id will be assigned to a group containing all of the stencils that comprise a given grob. For example, in the svg backend, the string will be assigned to the **id** attribute of a group (<g>) that encloses the stencils that comprise the grob. In the Postscript backend, as there is no way to group items, the setting of the id property will have no effect.

**ignore-ambitus** (boolean)

If set, don’t consider this notehead for ambitus calculation.

**ignore-collision** (boolean)

If set, don’t do note collision resolution on this **NoteColumn**.

**implicit** (boolean)

Is this an implicit bass figure?

**inspect-index** (integer)

If debugging is set, set beam and slur configuration to this index, and print the respective scores.

**inspect-quants** (pair of numbers)

If debugging is set, set beam and slur quants to this position, and print the respective scores.

**keep-inside-line** (boolean)

If set, this column cannot have objects sticking into the margin.

**kern** (dimension, in staff space)

The space between individual elements in any compound bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to **Staff.StaffSymbol.thickness**).

**knee** (boolean)

Is this beam kneed?

**knee-spacing-correction** (number)

Factor for the optical correction amount for kneed beams. Set between 0 for no correction and 1 for full correction.

**knee-to-beam** (boolean)

Determines whether a tuplet number will be positioned next to a kneed beam.

**labels** (list)

List of labels (symbols) placed on a column.

**layer** (integer)

An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

**ledger-extra** (dimension, in staff space)

Extra distance from staff line to draw ledger lines for.

**ledger-line-thickness** (pair of numbers)

The thickness of ledger lines. It is the sum of 2 numbers: The first is the factor for line thickness, and the second for staff space. Both contributions are added.

**ledger-positions** (list)

Repeating pattern for the vertical positions of ledger lines. Bracketed groups are always shown together.

**left-bound-info** (list)

An alist of properties for determining attachments of spanners to edges.

**left-padding** (dimension, in staff space)

The amount of space that is put left to an object (e.g., a lyric extender).

**length** (dimension, in staff space)

User override for the stem length of unbeamed stems.

**length-fraction** (number)

Multiplier for lengths. Used for determining ledger lines and stem lengths.

**line-break-penalty** (number)

Penalty for a line break at this column. This affects the choices of the line breaker; it avoids a line break at a column with a positive penalty and prefers a line break at a column with a negative penalty.

**line-break-permission** (symbol)

Instructs the line breaker on whether to put a line break at this column. Can be *force* or *allow*.

**line-break-system-details** (list)

An alist of properties to use if this column is the start of a system.

**line-count** (integer)

The number of staff lines.

**line-positions** (list)

Vertical positions of staff lines.

**line-thickness** (number)

For slurs and ties, this is the diameter of the virtual “pen” that draws the two arcs of the curve’s outline, which intersect at the endpoints. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to *Staff.StaffSymbol.thickness*).

**long-text** (markup)

Text markup. See Sezione “Formatting text” in *Guida alla Notazione*.

**max-beam-connect** (integer)

Maximum number of beams to connect to beams from this stem. Further beams are typeset as beamlets.

**maximum-gap** (number)

Maximum value allowed for *gap* property.

**measure-count** (integer)

The number of measures for a multi-measure rest.

**measure-length** (moment)

Length of a measure. Used in some spacing situations.

**merge-differently-dotted** (boolean)

Merge note heads in collisions, even if they have a different number of dots. This is normal notation for some types of polyphonic music.

**merge-differently-dotted** only applies to opposing stem directions (i.e., voice 1 & 2).

**merge-differently-headed** (boolean)

Merge note heads in collisions, even if they have different note heads. The smaller of the two heads is rendered invisible. This is used in polyphonic guitar notation. The value of this setting is used by Sezione “note-collision-interface” in *Guida al Funzionamento Interno*.

**merge-differently-headed** only applies to opposing stem directions (i.e., voice 1 & 2).

**minimum-distance** (dimension, in staff space)

Minimum distance between rest and notes or beam.

**minimum-length** (dimension, in staff space)

Try to make a spanner at least this long, normally in the horizontal direction. This requires an appropriate callback for the **springs-and-rods** property. If added to a **Tie**, this sets the minimum distance between noteheads.

**minimum-length-after-break** (dimension, in staff space)

If set, try to make a broken spanner starting a line this long. This requires an appropriate callback for the **springs-and-rods** property. If added to a **Tie**, this sets the minimum distance to the notehead.

**minimum-length-fraction** (number)

Minimum length of ledger line as fraction of note head size.

**minimum-space** (dimension, in staff space)

Minimum distance that the victim should move (after padding).

**minimum-X-extent** (pair of numbers)

Minimum size of an object in X dimension, measured in **staff-space** units.

**minimum-Y-extent** (pair of numbers)

Minimum size of an object in Y dimension, measured in **staff-space** units.

**neutral-direction** (direction)

Which direction to take in the center of the staff.

**neutral-position** (number)

Position (in half staff spaces) where to flip the direction of custos stem.

**next** (graphical (layout) object)

Object that is next relation (e.g., the lyric syllable following an extender).

**no-alignment** (boolean)

If set, don't place this grob in a **VerticalAlignment**; rather, place it using its own **Y-offset** callback.

**no-ledgers** (boolean)

If set, don't draw ledger lines on this object.

**no-stem-extend** (boolean)

If set, notes with ledger lines do not get stems extending to the middle staff line.

**non-break-align-symbols** (list)

A list of symbols that determine which NON-break-aligned interfaces to align this to.

**non-default** (boolean)

Set for manually specified clefs.

**non-musical** (boolean)

True if the grob belongs to a `NonMusicalPaperColumn`.

**nonstaff-nonstaff-spacing** (list)

The spacing alist controlling the distance between the current non-staff line and the next non-staff line in the direction of **staff-affinity**, if both are on the same side of the related staff, and **staff-affinity** is either UP or DOWN. See **staff-staff-spacing** for a description of the alist structure.

**nonstaff-relatedstaff-spacing** (list)

The spacing alist controlling the distance between the current non-staff line and the nearest staff in the direction of **staff-affinity**, if there are no non-staff lines between the two, and **staff-affinity** is either UP or DOWN. If **staff-affinity** is CENTER, then **nonstaff-relatedstaff-spacing** is used for the nearest staves on *both* sides, even if other non-staff lines appear between the current one and either of the staves. See **staff-staff-spacing** for a description of the alist structure.

**nonstaff-unrelatedstaff-spacing** (list)

The spacing alist controlling the distance between the current non-staff line and the nearest staff in the opposite direction from **staff-affinity**, if there are no other non-staff lines between the two, and **staff-affinity** is either UP or DOWN. See **staff-staff-spacing** for a description of the alist structure.

**normalized-endpoints** (pair)

Represents left and right placement over the total spanner, where the width of the spanner is normalized between 0 and 1.

**note-collision-threshold** (dimension, in staff space)

Simultaneous notes that are this close or closer in units of **staff-space** will be identified as vertically colliding. Used by **Stem** grobs for notes in the same voice, and **NoteCollision** grobs for notes in different voices. Default value 1.

**note-names** (vector)

Vector of strings containing names for easy-notation note heads.

**number-type** (symbol)

Numbering style. Choices include **roman-lower**, **roman-upper** and **arabic**.

**outside-staff-horizontal-padding** (number)

By default, an outside-staff-object can be placed so that is it very close to another grob horizontally. If this property is set, the outside-staff-object is raised so that it is not so close to its neighbor.

**outside-staff-padding** (number)

The padding to place between grobs when spacing according to **outside-staff-priority**. Two grobs with different **outside-staff-padding** values have the larger value of padding between them.

**outside-staff-placement-directive** (symbol)

One of four directives telling how outside staff objects should be placed.

- **left-to-right-greedy** – Place each successive grob from left to right.
- **left-to-right-polite** – Place a grob from left to right only if it does not potentially overlap with another grob that has been placed on a pass through a grob array. If there is overlap, do another pass to determine placement.
- **right-to-left-greedy** – Same as **left-to-right-greedy**, but from right to left.
- **right-to-left-polite** – Same as **left-to-right-polite**, but from right to left.

**outside-staff-priority** (number)

If set, the grob is positioned outside the staff in such a way as to avoid all collisions. In case of a potential collision, the grob with the smaller **outside-staff-priority** is closer to the staff.

**packed-spacing** (boolean)

If set, the notes are spaced as tightly as possible.

**padding** (dimension, in staff space)

Add this much extra space between objects that are next to each other.

**padding-pairs** (list)

An alist mapping (*name* . *name*) to distances.

**page-break-penalty** (number)

Penalty for page break at this column. This affects the choices of the page breaker; it avoids a page break at a column with a positive penalty and prefers a page break at a column with a negative penalty.

**page-break-permission** (symbol)

Instructs the page breaker on whether to put a page break at this column. Can be **force** or **allow**.

**page-turn-penalty** (number)

Penalty for a page turn at this column. This affects the choices of the page breaker; it avoids a page turn at a column with a positive penalty and prefers a page turn at a column with a negative penalty.

**page-turn-permission** (symbol)

Instructs the page breaker on whether to put a page turn at this column. Can be **force** or **allow**.

**parent-alignment-X** (number)

Specify on which point of the parent the object is aligned. The value **-1** means aligned on parent's left edge, **0** on center, and **1** right edge, in X direction. Other numerical values may also be specified - the unit is half the parent's width. If unset, the value from **self-alignment-X** property will be used.

**parent-alignment-Y** (number)

Like **parent-alignment-X** but for the Y axis.

**parenthesis-friends** (list)

A list of Grob types, as symbols. When parentheses enclose a Grob that has **'parenthesis-friends**, the parentheses widen to include any child Grobs with type among **'parenthesis-friends**.

- parenthesized** (boolean)  
Parenthesize this grob.
- positions** (pair of numbers)  
Pair of staff coordinates (*left . right*), where both *left* and *right* are in **staff-space** units of the current staff. For slurs, this value selects which slur candidate to use; if extreme positions are requested, the closest one is taken.
- prefer-dotted-right** (boolean)  
For note collisions, prefer to shift dotted up-note to the right, rather than shifting just the dot.
- protrusion** (number)  
In an arpeggio bracket, the length of the horizontal edges.
- ratio** (number)  
Parameter for slur shape. The higher this number, the quicker the slur attains its **height-limit**.
- remove-empty** (boolean)  
If set, remove group if it contains no interesting items.
- remove-first** (boolean)  
Remove the first staff of an orchestral score?
- remove-layer** (integer)  
The **Keep\_alive\_together\_engraver** removes all **VerticalAxisGroup** grobs with a **remove-layer** larger than the smallest retained **remove-layer**. Set to **#f** to make a layer invisible to the **Keep\_alive\_together\_engraver**, set to '()' to have it not participate in the layering decisions.
- replacement-alist** (list)  
Alist of strings. The key is a string of the pattern to be replaced. The value is a string of what should be displayed. Useful for ligatures.
- restore-first** (boolean)  
Print a natural before the accidental.
- rhythmic-location** (rhythmic location)  
Where (bar number, measure position) in the score.
- right-bound-info** (list)  
An alist of properties for determining attachments of spanners to edges.
- right-padding** (dimension, in staff space)  
Space to insert on the right side of an object (e.g., between note and its accidentals).
- rotation** (list)  
Number of degrees to rotate this object, and what point to rotate around. For example, '(45 0 0) rotates by 45 degrees around the center of this object.
- round-up-exceptions** (list)  
A list of pairs where car is the numerator and cdr the denominator of a moment. Each pair in this list means that the multi-measure rests of the corresponding length will be rounded up to the longer rest. See *round-up-to-longer-rest*.
- round-up-to-longer-rest** (boolean)  
Displays the longer multi-measure rest when the length of a measure is between two values of **usable-duration-logs**. For example, displays a breve instead of a whole in a 3/2 measure.

**rounded** (boolean)

Decide whether lines should be drawn rounded or not.

**same-direction-correction** (number)

Optical correction amount for stems that are placed in tight configurations. This amount is used for stems with the same direction to compensate for note head to stem distance.

**script-priority** (number)

A key for determining the order of scripts in a stack, by being added to the position of the script in the user input, the sum being the overall priority. Smaller means closer to the head.

**segno-kern** (number)

The space between the two thin lines of the segno bar line symbol, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to `Staff.StaffSymbol.thickness`).

**self-alignment-X** (number)

Specify alignment of an object. The value -1 means left aligned, 0 centered, and 1 right-aligned in X direction. Other numerical values may also be specified - the unit is half the object width.

**self-alignment-Y** (number)

Like **self-alignment-X** but for the Y axis.

**shape** (symbol)

This setting determines what shape a grob has. Valid choices depend on the `stencil` callback reading this property.

**sharp-positions** (list)

Sharps in key signatures are placed within the specified ranges of staff-positions. The general form is a list of pairs, with one pair for each type of clef, in order of the staff-position at which each clef places C: (**alto treble tenor soprano baritone mezzosoprano bass**). If the list contains a single element it applies for all clefs. A single number in place of a pair sets accidentals within the octave ending at that staff-position.

**shorten-pair** (pair of numbers)

The lengths to shorten a text-spanner on both sides, for example a pedal bracket. Positive values shorten the text-spanner, while negative values lengthen it.

**shortest-duration-space** (number)

Start with this multiple of **spacing-increment** space for the shortest duration. See also Sezione “spacing-spanner-interface” in *Guida al Funzionamento Interno*.

**shortest-playing-duration** (moment)

The duration of the shortest note playing here.

**shortest-starter-duration** (moment)

The duration of the shortest note that starts here.

**side-axis** (number)

If the value is X (or equivalently 0), the object is placed horizontally next to the other object. If the value is Y or 1, it is placed vertically.

**side-relative-direction** (direction)

Multiply direction of **direction-source** with this to get the direction of this object.



**simple-Y** (boolean)

Should the Y placement of a spanner disregard changes in system heights?

**size** (number)

The ratio of the size of the object to its default size.

**skip-quanting** (boolean)

Should beam quanting be skipped?

**skyline-horizontal-padding** (number)

For determining the vertical distance between two staves, it is possible to have a configuration which would result in a tight interleaving of grobs from the top staff and the bottom staff. The larger this parameter is, the farther apart the staves are placed in such a configuration.

**skyline-vertical-padding** (number)

The amount by which the left and right skylines of a column are padded vertically, beyond the **Y-extents** and **extra-spacing-heights** of the constituent grobs in the column. Increase this to prevent interleaving of grobs from adjacent columns.

**slash-negative-kern** (number)

The space to remove between slashes in percent repeat glyphs. Larger values bring the two elements closer together.

**slope** (number)

The slope of this object.

**slur-padding** (number)

Extra distance between slur and script.

**snap-radius** (number)

The maximum distance between two objects that will cause them to snap to alignment along an axis.

**space-alist** (list)

An alist that specifies distances from this grob to other breakable items, using the format:

```
'((break-align-symbol . (spacing-style . space))
  (break-align-symbol . (spacing-style . space))
  ...)
```

Standard choices for *break-align-symbol* are listed in Sezione “break-alignment-interface” in *Guida al Funzionamento Interno*. Additionally, three special break-align symbols available to **space-alist** are:

**first-note**

used when the grob is just left of the first note on a line

**next-note**

used when the grob is just left of any other note; if not set, the value of **first-note** gets used

**right-edge**

used when the grob is the last item on the line (only compatible with the **extra-space** spacing style)

Choices for *spacing-style* are:

**extra-space**

Put this much space between the two grobs. The space is stretchable when paired with **first-note** or **next-note**; otherwise it is fixed.

**minimum-space**

Put at least this much space between the left sides of both grobs, without allowing them to collide. The space is stretchable when paired with **first-note** or **next-note**; otherwise it is fixed. Not compatible with **right-edge**.

**fixed-space**

Only compatible with **first-note** and **next-note**. Put this much fixed space between the grob and the note.

**minimum-fixed-space**

Only compatible with **first-note** and **next-note**. Put at least this much fixed space between the left side of the grob and the left side of the note, without allowing them to collide.

**semi-fixed-space**

Only compatible with **first-note** and **next-note**. Put this much space between the grob and the note, such that half of the space is fixed and half is stretchable.

Rules for this spacing are much more complicated than this. See [Wanske] page 126–134, [Ross] page 143–147.

**space-to-barline** (boolean)

If set, the distance between a note and the following non-musical column will be measured to the bar line instead of to the beginning of the non-musical column. If there is a clef change followed by a bar line, for example, this means that we will try to space the non-musical column as though the clef is not there.

**spacing-increment** (dimension, in staff space)

The unit of length for note-spacing. Typically, the width of a note head. See also Sezione “spacing-spanner-interface” in *Guida al Funzionamento Interno*.

**spacing-pair** (pair)

A pair of alignment symbols which set an object’s spacing relative to its left and right **BreakAlignments**.

For example, a **MultiMeasureRest** will ignore prefatory items at its bounds (i.e., clefs, key signatures and time signatures) using the following override:

```
\override MultiMeasureRest
  #'spacing-pair = #'(staff-bar . staff-bar)
```

**spanner-id** (string)

An identifier to distinguish concurrent spanners.

**springs-and-rods** (boolean)

Dummy variable for triggering spacing routines.

**stacking-dir** (direction)

Stack objects in which direction?

**staff-affinity** (direction)

The direction of the staff to use for spacing the current non-staff line. Choices are **UP**, **DOWN**, and **CENTER**. If **CENTER**, the non-staff line will be placed equidistant between the two nearest staves on either side, unless collisions or other spacing constraints prevent this. Setting **staff-affinity** for a staff causes it to be treated as a non-staff line. Setting **staff-affinity** to **#f** causes a non-staff line to be treated as a staff.

**staff-padding** (dimension, in staff space)

Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics **p** and **f**) on their baselines.

**staff-position** (number)

Vertical position, measured in half staff spaces, counted from the middle line.

**staff-space** (dimension, in staff space)

Amount of space between staff lines, expressed in global **staff-space**.

**staff-staff-spacing** (list)

When applied to a staff-group's **StaffGrouper** grob, this spacing alist controls the distance between consecutive staves within the staff-group. When applied to a staff's **VerticalAxisGroup** grob, it controls the distance between the staff and the nearest staff below it in the same system, replacing any settings inherited from the **StaffGrouper** grob of the containing staff-group, if there is one. This property remains in effect even when non-staff lines appear between staves. The alist can contain the following keys:

- **basic-distance** – the vertical distance, measured in staff-spaces, between the reference points of the two items when no collisions would result, and no stretching or compressing is in effect.
- **minimum-distance** – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect.
- **padding** – the minimum required amount of unobstructed vertical whitespace between the bounding boxes (or skylines) of the two items, measured in staff-spaces.
- **stretchability** – a unitless measure of the dimension's relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result).

**staffgroup-staff-spacing** (list)

The spacing alist controlling the distance between the last staff of the current staff-group and the staff just below it in the same system, even if one or more non-staff lines exist between the two staves. If the **staff-staff-spacing** property of the staff's **VerticalAxisGroup** grob is set, that is used instead. See **staff-staff-spacing** for a description of the alist structure.

**stem-attachment** (pair of numbers)

An (x . y) pair where the stem attaches to the notehead.

**stem-begin-position** (number)

User override for the begin position of a stem.

**stem-spacing-correction** (number)

Optical correction amount for stems that are placed in tight configurations. For opposite directions, this amount is the correction for two normal sized stems that overlap completely.

**stemlet-length** (number)

How long should be a stem over a rest?

**stencil** (stencil)

The symbol to print.

**stencils** (list)

Multiple stencils, used as intermediate value.

**strict-grace-spacing** (boolean)

If set, main notes are spaced normally, then grace notes are put left of the musical columns for the main notes.

**strict-note-spacing** (boolean)

If set, unbroken columns with non-musical material (clefs, bar lines, etc.) are not spaced separately, but put before musical columns.

**stroke-style** (string)

Set to "grace" to turn stroke through flag on.

**style** (symbol)

This setting determines in what style a grob is typeset. Valid choices depend on the **stencil** callback reading this property.

**text** (markup)

Text markup. See Sezione "Formatting text" in *Guida alla Notazione*.

**text-direction** (direction)

This controls the ordering of the words. The default **RIGHT** is for roman text. Arabic or Hebrew should use **LEFT**.

**thick-thickness** (number)

Thickness of the thick line in a bar line, expressed as a multiple of the default staff-line thickness (i.e. the visual output is *not* influenced by changes to **Staff.StaffSymbol.thickness**).

**thickness** (number)

For grobs made up of lines, this is the thickness of the line. For slurs and ties, this is the distance between the two arcs of the curve's outline at its thickest point, not counting the diameter of the virtual "pen" that draws the arcs. This property is expressed as a multiple of the current staff-line thickness (i.e. the visual output is influenced by changes to **Staff.StaffSymbol.thickness**).

**tie-configuration** (list)

List of (**position** . **dir**) pairs, indicating the desired tie configuration, where **position** is the offset from the center of the staff in staff space and **dir** indicates the direction of the tie (1=>up, -1=>down, 0=>center). A non-pair entry in the list causes the corresponding tie to be formatted automatically.

**to-barline** (boolean)

If true, the spanner will stop at the bar line just before it would otherwise stop.

**toward-stem-shift** (number)

Amount by which scripts are shifted toward the stem if their direction coincides with the stem direction. 0.0 means centered on the note head (the default position of most scripts); 1.0 means centered on the stem. Interpolated values are possible.

**toward-stem-shift-in-column** (number)

Amount by which a script is shifted toward the stem if its direction coincides with the stem direction and it is associated with a **ScriptColumn** object. 0.0 means centered on the note head (the default position of most scripts); 1.0 means centered on the stem. Interpolated values are possible.

**transparent** (boolean)

This makes the grob invisible.

**uniform-stretching** (boolean)

If set, items stretch proportionally to their natural separation based on durations. This looks better in complex polyphonic patterns.

**usable-duration-logs** (list)

List of **duration-logs** that can be used in typesetting the grob.

**use-skylines** (boolean)

Should skylines be used for side positioning?

**used** (boolean)

If set, this spacing column is kept in the spacing problem.

**vertical-skylines** (pair of skylines)

Two skylines, one above and one below this grob.

**voiced-position** (number)

The staff-position of a voiced **Rest**, negative if the rest has **direction DOWN**.

**when** (moment)

Global time step associated with this column.

**whiteout** (boolean-or-number)

If a number or true, the grob is printed over a white background to white-out underlying material, if the grob is visible. A number indicates how far the white background extends beyond the bounding box of the grob as a multiple of the staff-line thickness. The shape of the background is determined by **whiteout-style**. Usually **#f** by default.

**whiteout-style** (symbol)

Determines the shape of the **whiteout** background. Available are 'outline, 'rounded-box, and the default 'box.

**width** (dimension, in staff space)

The width of a grob measured in staff space.

**word-space** (dimension, in staff space)

Space to insert between words in texts.

**X-align-on-main-noteheads** (boolean)

If true, this grob will ignore suspended noteheads when aligning itself on **NoteColumn**.

**X-extent** (pair of numbers)

Extent (size) in the X direction, measured in staff-space units, relative to object's reference point.

**X-offset** (number)

The horizontal amount that this object is moved relative to its X-parent.

**X-positions** (pair of numbers)

Pair of X staff coordinates of a spanner in the form (*left* . *right*), where both *left* and *right* are in **staff-space** units of the current staff.

**Y-extent** (pair of numbers)

Extent (size) in the Y direction, measured in staff-space units, relative to object's reference point.

**Y-offset** (number)

The vertical amount that this object is moved relative to its Y-parent.

**zigzag-length** (dimension, in staff space)

The length of the lines of a zigzag, relative to **zigzag-width**. A value of 1 gives 60-degree zigzags.

**zigzag-width** (dimension, in staff space)

The width of one zigzag squiggle. This number is adjusted slightly so that the glissando line can be constructed from a whole number of squiggles.

## A.19 Funzioni musicali disponibili

**absolute** [music] - *music* (music)

Make *music* absolute. This does not actually change the music itself but rather hides it from surrounding `\relative` and `\fixed` commands.

**acciaccatura** [music] - *music* (music)

Create an acciaccatura from the following music expression

**accidentalStyle** [music] - *style* (symbol list)

Set accidental style to symbol list *style* in the form ‘`piano-cautionary`’. If *style* has a form like ‘`Staff.piano-cautionary`’, the settings are applied to that context. Otherwise, the context defaults to ‘`Staff`’, except for piano styles, which use ‘`GrandStaff`’ as a context.

**addChordShape** [void] - *key-symbol* (symbol) *tuning* (pair) *shape-definition* (string or pair)

Add chord shape *shape-definition* to the *chord-shape-table* hash with the key (`cons key-symbol tuning`).

**addInstrumentDefinition** [void] - *name* (string) *lst* (list)

Create instrument *name* with properties *list*.

**addQuote** [void] - *name* (string) *music* (music)

Define *music* as a quotable music expression named *name*

**afterGrace** [music] - *main* (music) *grace* (music)

Create *grace* note(s) after a *main* music expression.

**allowPageTurn** [music]

Allow a page turn. May be used at toplevel (ie between scores or markups), or inside a score.

**allowVoltaHook** [void] - *bar* (string)

(undocumented; fixme)

**alterBroken** [music] - *property* (symbol list or symbol) *arg* (list) *item* (symbol list or music)

Override *property* for pieces of broken spanner *item* with values *arg*. *item* may either be music in the form of a starting spanner event, or a symbol list in the form ‘`Context.Grob`’ or just ‘`Grob`’. If *item* is in the form of a spanner event, *property* may also have the form ‘`Grob.property`’ for specifying a directed tweak.

**appendToTag** [music] - *tag* (symbol) *more* (music) *music* (music)

Append *more* to the `elements` of all music expressions in *music* that are tagged with *tag*.

**applyContext** [music] - *proc* (procedure)

Modify context properties with Scheme procedure *proc*.

**applyMusic** [music] - *func* (procedure) *music* (music)

Apply procedure *func* to *music*.

**applyOutput** [music] - *target* (symbol list or symbol) *proc* (procedure)

Apply function *proc* to every layout object matched by *target* which takes the form `Context` or `Context.Grob`.

**appoggiatura** [music] - *music* (music)

Create an appoggiatura from *music*

**assertBeamQuant** [music] - *l* (pair) *r* (pair)

Testing function: check whether the beam quants *l* and *r* are correct

**assertBeamSlope** [music] - *comp* (procedure)

Testing function: check whether the slope of the beam is the same as *comp*

**autochange** [music] - *pitch* [pitch] *clef-1* [context modification] *clef-2* [context modification]  
*music* (music)

Make voices that switch between staves automatically. As an option the pitch where to switch staves may be specified. The clefs for the staves are optional as well. Setting clefs works only for implicitly instantiated staves.

**balloonGrobText** [music] - *grob-name* (symbol) *offset* (pair of numbers) *text* (markup)

Attach *text* to *grob-name* at offset *offset* (use like `\once`)

**balloonText** [post event] - *offset* (pair of numbers) *text* (markup)

Attach *text* at *offset* (use like `\tweak`)

**bar** [music] - *type* (string)

Insert a bar line of type *type*

**barNumberCheck** [music] - *n* (integer)

Print a warning if the current bar number is not *n*.

**beamExceptions** (any type) - *music* (music)

Extract a value suitable for setting `Timing.beamExceptions` from the given pattern with explicit beams in *music*. A bar check `|` has to be used between bars of patterns in order to reset the timing.

**bendAfter** [post event] - *delta* (real number)

Create a fall or doit of pitch interval *delta*.

**bookOutputName** [void] - *newfilename* (string)

Direct output for the current book block to *newfilename*.

**bookOutputSuffix** [void] - *newsuffix* (string)

Set the output filename suffix for the current book block to *newsuffix*.

**breathe** [music]

Insert a breath mark.

**chordRepeats** [music] - *event-types* [list] *music* (music)

Walk through *music* putting the notes of the previous chord into repeat chords, as well as an optional list of *event-types* such as `#'(string-number-event)`.

**clef** [music] - *type* (string)

Set the current clef to *type*.

**compoundMeter** [music] - *args* (pair)

Create compound time signatures. The argument is a Scheme list of lists. Each list describes one fraction, with the last entry being the denominator, while the first entries describe the summands in the numerator. If the time signature consists of just one fraction, the list can be given directly, i.e. not as a list containing a single list. For example, a time signature of  $(3+1)/8 + 2/4$  would be created as `\compoundMeter #'((3 1 8) (2 4))`, and a time signature of  $(3+2)/8$  as `\compoundMeter #'((3 2 8))` or shorter `\compoundMeter #'(3 2 8)`.

**compressMMRests** [music] - *music* (music)

Remove the empty bars created by multi-measure rests, leaving just the first bar containing the MM rest itself.

- crossStaff** [music] - *notes* (music)  
Create cross-staff stems
- cueClef** [music] - *type* (string)  
Set the current cue clef to *type*.
- cueClefUnset** [music]  
Unset the current cue clef.
- cueDuring** [music] - *what* (string) *dir* (direction) *main-music* (music)  
Insert contents of quote *what* corresponding to *main-music*, in a CueVoice oriented by *dir*.
- cueDuringWithClef** [music] - *what* (string) *dir* (direction) *clef* (string) *main-music* (music)  
Insert contents of quote *what* corresponding to *main-music*, in a CueVoice oriented by *dir*.
- deadNote** [music] - *note* (music)  
Print *note* with a cross-shaped note head.
- defaultNoteHeads** [music]  
Revert to the default note head style.
- defineBarLine** [void] - *bar* (string) *glyph-list* (list)  
Define bar line settings for bar line *bar*. The list *glyph-list* must have three entries which define the appearance at the end of line, at the beginning of the next line, and the span bar, respectively.
- displayLilyMusic** [music] - *port* [output port] *music* (music)  
Display the LilyPond input representation of *music* to *port*, defaulting to the console.
- displayMusic** [music] - *port* [output port] *music* (music)  
Display the internal representation of *music* to *port*, default to the console.
- displayScheme** (any type) - *port* [output port] *expr* (any type)  
Display the internal representation of *expr* to *port*, default to the console.
- endSpanners** [music] - *music* (music)  
Terminate the next spanner prematurely after exactly one note without the need of a specific end spanner.
- eventChords** [music] - *music* (music)  
Compatibility function wrapping **EventChord** around isolated rhythmic events occurring since version 2.15.28, after expanding repeat chords ‘q’.
- featherDurations** [music] - *factor* (moment) *argument* (music)  
Adjust durations of music in *argument* by rational *factor*.
- finger** [post event] - *finger* (number or markup)  
Apply *finger* as a fingering indication.
- fixed** [music] - *pitch* (pitch) *music* (music)  
Use the octave of *pitch* as the default octave for *music*.
- footnote** [music] - *mark* [markup] *offset* (pair of numbers) *footnote* (markup) *item* (symbol list or music)  
Make the markup *footnote* a footnote on *item*. The footnote is marked with a markup *mark* moved by *offset* with respect to the marked music.  
If *mark* is not given or specified as `\default`, it is replaced by an automatically generated sequence number. If *item* is a symbol list of form ‘Grob’ or ‘Context.Grob’,



then grobs of that type will be marked at the current time step in the given context (default **Bottom**).

If *item* is music, the music will get a footnote attached to a grob immediately attached to the event, like `\tweak` does. For attaching a footnote to an *indirectly* caused grob, write `\single\footnote`, use *item* to specify the grob, and follow it with the music to annotate.

Like with `\tweak`, if you use a footnote on a following post-event, the `\footnote` command itself needs to be attached to the preceding note or rest as a post-event with `-`.

**grace** [music] - *music* (music)

Insert *music* as grace notes.

**grobdescriptions** (any type) - *descriptions* (list)

Create a context modification from *descriptions*, a list in the format of **all-grob-descriptions**.

**harmonicByFret** [music] - *fret* (number) *music* (music)

Convert *music* into mixed harmonics; the resulting notes resemble harmonics played on a fretted instrument by touching the strings at *fret*.

**harmonicByRatio** [music] - *ratio* (number) *music* (music)

Convert *music* into mixed harmonics; the resulting notes resemble harmonics played on a fretted instrument by touching the strings at the point given through *ratio*.

**harmonicNote** [music] - *note* (music)

Print *note* with a diamond-shaped note head.

**harmonicsOn** [music]

Set the default note head style to a diamond-shaped style.

**hide** [music] - *item* (symbol list or music)

Set *item*'s 'transparent' property to `#t`, making it invisible while still retaining its dimensions.

If *item* is a symbol list of form **GrobName** or **Context.GrobName**, the result is an override for the grob name specified by it. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied to it.

**incipit** [music] - *incipit-music* (music)

Output *incipit-music* before the main staff as an indication of its appearance in the original music.

**inherit-acceptability** [void] - *to* (symbol) *from* (symbol)

When used in an output definition, will modify all context definitions such that context *to* is accepted as a child by all contexts that also accept *from*.

**inStaffSegno** [music]

Put the segno variant 'varsegno' at this position into the staff, compatible with the repeat command.

**instrumentSwitch** [music] - *name* (string)

Switch instrument to *name*, which must be predefined with `\addInstrumentDefinition`.

**inversion** [music] - *around* (pitch) *to* (pitch) *music* (music)

Invert *music* about *around* and transpose from *around* to *to*.

- keepWithTag** [music] - *tags* (symbol list or symbol) *music* (music)  
 Include only elements of *music* that are tagged with one of the tags in *tags*. *tags* may be either a single symbol or a list of symbols.  
 Each tag may be declared as a member of at most one tag group (defined with `\tagGroup`). If none of a *music* element's tags share a tag group with one of the specified *tags*, the element is retained.
- key** [music] - *tonic* [pitch] *pitch-alist* [list]  
 Set key to *tonic* and scale *pitch-alist*. If both are null, just generate `KeyChangeEvent`.
- killCues** [music] - *music* (music)  
 Remove cue notes from *music*.
- label** [music] - *label* (symbol)  
 Create *label* as a bookmarking label.
- language** [void] - *language* (string)  
 Set note names for language *language*.
- languageRestore** [void]  
 Restore a previously-saved pitchnames alist.
- languageSaveAndChange** [void] - *language* (string)  
 Store the previous pitchnames alist, and set a new one.
- magnifyMusic** [music] - *mag* (positive number) *music* (music)  
 Magnify the notation of *music* without changing the staff-size, using *mag* as a size factor. Stems, beams, slurs, ties, and horizontal spacing are adjusted automatically.
- magnifyStaff** [music] - *mag* (positive number)  
 Change the size of the staff, adjusting notation size and horizontal spacing automatically, using *mag* as a size factor.
- makeClusters** [music] - *arg* (music)  
 Display chords in *arg* as clusters.
- makeDefaultStringTuning** [void] - *symbol* (symbol) *pitches* (list)  
 This defines a string tuning *symbol* via a list of *pitches*. The *symbol* also gets registered in `defaultStringTunings` for documentation purposes.
- mark** [music] - *label* [number or markup]  
 Make the music for the `\mark` command.
- markupMap** [music] - *path* (symbol list or symbol) *markupfun* (markup-function) *music* (music)  
 This applies the given markup function *markupfun* to all markup music properties matching *path* in *music*.  
 For example,  

```
\new Voice { g'2 c'' }
\addlyrics {
  \markupMap LyricEvent.text
    \markup \with-color #red \etc
    { Oh yes! }
}
```
- modalInversion** [music] - *around* (pitch) *to* (pitch) *scale* (music) *music* (music)  
 Invert *music* about *around* using *scale* and transpose from *around* to *to*.
- modalTranspose** [music] - *from* (pitch) *to* (pitch) *scale* (music) *music* (music)  
 Transpose *music* from pitch *from* to pitch *to* using *scale*.

- musicMap** [music] - *proc* (procedure) *mus* (music)  
Apply *proc* to *mus* and all of the music it contains.
- noPageBreak** [music]  
Forbid a page break. May be used at toplevel (i.e., between scores or markups), or inside a score.
- noPageTurn** [music]  
Forbid a page turn. May be used at toplevel (i.e., between scores or markups), or inside a score.
- octaveCheck** [music] - *pitch* (pitch)  
Octave check.
- offset** [music] - *property* (symbol list or symbol) *offsets* (any type) *item* (symbol list or music)  
Offset the default value of *property* of *item* by *offsets*. If *item* is a string, the result is `\override` for the specified grob type. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied.
- omit** [music] - *item* (symbol list or music)  
Set *item*'s 'stencil' property to `#f`, effectively omitting it without taking up space.  
If *item* is a symbol list of form `GrobName` or `Context.GrobName`, the result is an override for the grob name specified by it. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied to it.
- once** [music] - *music* (music)  
Set *once* to `#t` on all layout instruction events in *music*. This will complain about music with an actual duration. As a special exception, if *music* contains 'tweaks' it will be silently ignored in order to allow for `\once \propertyTweak` to work as both one-time override and proper tweak.
- ottava** [music] - *octave* (integer)  
Set the octavation.
- overrideProperty** [music] - *grob-property-path* (symbol list) *value* (any type)  
Set the grob property specified by *grob-property-path* to *value*. *grob-property-path* is a symbol list of the form `Context.GrobName.property` or `GrobName.property`, possibly with subproperties given as well.  
As opposed to `\override` which overrides the context-dependent defaults with which a grob is created, this command uses `Output_property_engraver` at the grob acknowledgment stage. This may be necessary for overriding values set after the initial grob creation.
- overrideTimeSignatureSettings** [music] - *time-signature* (fraction, as pair) *base-moment* (fraction, as pair) *beat-structure* (list) *beam-exceptions* (list)  
Override `timeSignatureSettings` for time signatures of *time-signature* to have settings of *base-moment*, *beat-structure*, and *beam-exceptions*.
- pageBreak** [music]  
Force a page break. May be used at toplevel (i.e., between scores or markups), or inside a score.
- pageTurn** [music]  
Force a page turn between two scores or top-level markups.
- palmMute** [music] - *note* (music)  
Print *note* with a triangle-shaped note head.

**palmMuteOn** [music]

Set the default note head style to a triangle-shaped style.

**parallelMusic** [void] - *voice-ids* (list) *music* (music)

Define parallel music sequences, separated by '|' (bar check signs), and assign them to the identifiers provided in *voice-ids*.

*voice-ids*: a list of music identifiers (symbols containing only letters)

*music*: a music sequence, containing BarChecks as limiting expressions.

Example:

```
\parallelMusic #'(A B C) {
  c c | d d | e e |
  d d | e e | f f |
}
<==>
A = { c c | d d }
B = { d d | e e }
C = { e e | f f }
```

The last bar checks in a sequence are not copied to the result in order to facilitate ending the last entry at non-bar boundaries.

**parenthesize** [music] - *arg* (music)

Tag *arg* to be parenthesized.

**partcombine** [music] - *chord-range* [pair of numbers] *part1* (music) *part2* (music)

Take the music in *part1* and *part2* and return a music expression containing simultaneous voices, where *part1* and *part2* are combined into one voice where appropriate. Optional *chord-range* sets the distance in steps between notes that may be combined into a chord or unison.

**partcombineDown** [music] - *chord-range* [pair of numbers] *part1* (music) *part2* (music)

Take the music in *part1* and *part2* and typeset so that they share a staff with stems directed downward.

**partcombineForce** [music] - *type* [symbol]

Override the part-combiner.

**partcombineUp** [music] - *chord-range* [pair of numbers] *part1* (music) *part2* (music)

Take the music in *part1* and *part2* and typeset so that they share a staff with stems directed upward.

**partial** [music] - *dur* (duration)

Make a partial measure.

**phrasingSlurDashPattern** [music] - *dash-fraction* (number) *dash-period* (number)

Set up a custom style of dash pattern for *dash-fraction* ratio of line to space repeated at *dash-period* interval for phrasing slurs.

**pitchedTrill** [music] - *main-note* (music) *secondary-note* (music)

Print a trill with *main-note* as the main note of the trill and print *secondary-note* as a stemless note head in parentheses.

**pointAndClickOff** [void]

Suppress generating extra code in final-format (e.g. pdf) files to point back to the lilypond source statement.

**pointAndClickOn** [void]

Enable generation of code in final-format (e.g. pdf) files to reference the originating lilypond source statement; this is helpful when developing a score but generates bigger final-format files.

**pointAndClickTypes** [void] - *types* (symbol list or symbol)

Set a type or list of types (such as `#'note-event`) for which point-and-click info is generated.

**propertyOverride** [music] - *grob-property-path* (symbol list) *value* (any type)

Set the grob property specified by *grob-property-path* to *value*. *grob-property-path* is a symbol list of the form `Context.GrobName.property` or `GrobName.property`, possibly with subproperties given as well. This music function is mostly intended for use from Scheme as a substitute for the built-in `\override` command.

**propertyRevert** [music] - *grob-property-path* (symbol list)

Revert the grob property specified by *grob-property-path* to its previous value. *grob-property-path* is a symbol list of the form `Context.GrobName.property` or `GrobName.property`, possibly with subproperties given as well. This music function is mostly intended for use from Scheme as a substitute for the built-in `\revert` command.

**propertySet** [music] - *property-path* (symbol list or symbol) *value* (any type)

Set the context property specified by *property-path* to *value*. This music function is mostly intended for use from Scheme as a substitute for the built-in `\set` command.

**propertyTweak** [music] - *prop* (symbol list or symbol) *value* (any type) *item* (symbol list or music)

Add a tweak to the following *item*, usually music. This generally behaves like `\tweak` but will turn into an `\override` when *item* is a symbol list.

In that case, *item* specifies the grob path to override. This is mainly useful when using `\propertyTweak` as a component for building other functions like `\omit`. It is not the default behavior for `\tweak` since many input strings in `\lyricmode` can serve equally as music or as symbols which causes surprising behavior when tweaking lyrics using the less specific semantics of `\propertyTweak`.

*prop* can contain additional elements in which case a nested property (inside of an alist) is tweaked.

**propertyUnset** [music] - *property-path* (symbol list or symbol)

Unset the context property specified by *property-path*. This music function is mostly intended for use from Scheme as a substitute for the built-in `\unset` command.

**pushToTag** [music] - *tag* (symbol) *more* (music) *music* (music)

Add *more* to the front of **elements** of all music expressions in *music* that are tagged with *tag*.

**quoteDuring** [music] - *what* (string) *main-music* (music)

Indicate a section of music to be quoted. *what* indicates the name of the quoted voice, as specified in an `\addQuote` command. *main-music* is used to indicate the length of music to be quoted; usually contains spacers or multi-measure rests.

**relative** [music] - *pitch* [pitch] *music* (music)

Make *music* relative to *pitch*. If *pitch* is omitted, the first note in *music* is given in absolute pitch.

**removeWithTag** [music] - *tags* (symbol list or symbol) *music* (music)

Remove elements of *music* that are tagged with one of the tags in *tags*. *tags* may be either a single symbol or a list of symbols.

- resetRelativeOctave** [music] - *pitch* (pitch)  
Set the octave inside a `\relative` section.
- retrograde** [music] - *music* (music)  
Return *music* in reverse order.
- revertTimeSignatureSettings** [music] - *time-signature* (pair)  
Revert **timeSignatureSettings** for time signatures of *time-signature*.
- rightHandFinger** [post event] - *finger* (number or markup)  
Apply *finger* as a fingering indication.
- scaleDurations** [music] - *fraction* (fraction, as pair) *music* (music)  
Multiply the duration of events in *music* by *fraction*.
- settingsFrom** (any type) - *ctx* [symbol] *music* (music)  
Take the layout instruction events from *music*, optionally restricted to those applying to context type *ctx*, and return a context modification duplicating their effect.
- shape** [music] - *offsets* (list) *item* (symbol list or music)  
Offset control-points of *item* by *offsets*. The argument is a list of number pairs or list of such lists. Each element of a pair represents an offset to one of the coordinates of a control-point. If *item* is a string, the result is `\once\override` for the specified grob type. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied.
- shiftDurations** [music] - *dur* (integer) *dots* (integer) *arg* (music)  
Change the duration of *arg* by adding *dur* to the `durlog` of *arg* and *dots* to the `dots` of *arg*.
- single** [music] - *overrides* (music) *music* (music)  
Convert *overrides* to tweaks and apply them to *music*. This does not convert `\revert`, `\set` or `\unset`.
- skip** [music] - *dur* (duration)  
Skip forward by *dur*.
- slashedGrace** [music] - *music* (music)  
Create slashed graces (slashes through stems, but no slur) from the following music expression
- slurDashPattern** [music] - *dash-fraction* (number) *dash-period* (number)  
Set up a custom style of dash pattern for *dash-fraction* ratio of line to space repeated at *dash-period* interval for slurs.
- spacingTweaks** [music] - *parameters* (list)  
Set the system stretch, by reading the 'system-stretch' property of the 'parameters' assoc list.
- storePredefinedDiagram** [void] - *fretboard-table* (hash table) *chord* (music) *tuning* (pair) *diagram-definition* (string or pair)  
Add predefined fret diagram defined by *diagram-definition* for the chord pitches *chord* and the stringTuning *tuning*.
- stringTuning** (any type) - *chord* (music)  
Convert *chord* to a string tuning. *chord* must be in absolute pitches and should have the highest string number (generally the lowest pitch) first.
- styledNoteHeads** [music] - *style* (symbol) *heads* (symbol list or symbol) *music* (music)  
Set *heads* in *music* to *style*.

**tabChordRepeats** [*music*] - *event-types* [list] *music* (music)

Walk through *music* putting the notes, fingerings and string numbers of the previous chord into repeat chords, as well as an optional list of *event-types* such as `#'(articulation-event)`.

**tabChordRepetition** [void]

Include the string and fingering information in a chord repetition. This function is deprecated; try using `\tabChordRepeats` instead.

**tag** [*music*] - *tags* (symbol list or symbol) *music* (music)

Tag the following *music* with *tags* and return the result, by adding the single symbol or symbol list *tags* to the **tags** property of *music*.

**tagGroup** [void] - *tags* (symbol list)

Define a tag group comprising the symbols in the symbol list *tags*. Tag groups must not overlap.

**temporary** [*music*] - *music* (music)

Make any `\override` in *music* replace an existing grob property value only temporarily, restoring the old value when a corresponding `\revert` is executed. This is achieved by clearing the ‘pop-first’ property normally set on `\overrides`.

An `\override/\revert` sequence created by using `\temporary` and `\undo` on the same music containing overrides will cancel out perfectly or cause a warning.

Non-property-related music is ignored, warnings are generated for any property-changing music that isn’t an `\override`.

**tieDashPattern** [*music*] - *dash-fraction* (number) *dash-period* (number)

Set up a custom style of dash pattern for *dash-fraction* ratio of line to space repeated at *dash-period* interval for ties.

**time** [*music*] - *beat-structure* [number list] *fraction* (fraction, as pair)

Set *fraction* as time signature, with optional number list *beat-structure* before it.

**times** [*music*] - *fraction* (fraction, as pair) *music* (music)

Scale *music* in time by *fraction*.

**tocItem** [*music*] - *text* (markup)

Add a line to the table of content, using the **tocItemMarkup** paper variable markup

**transpose** [*music*] - *from* (pitch) *to* (pitch) *music* (music)

Transpose *music* from pitch *from* to pitch *to*.

**transposedCueDuring** [*music*] - *what* (string) *dir* (direction) *pitch* (pitch) *main-music* (music)

Insert notes from the part *what* into a voice called **cue**, using the transposition defined by *pitch*. This happens simultaneously with *main-music*, which is usually a rest. The argument *dir* determines whether the cue notes should be notated as a first or second voice.

**transposition** [*music*] - *pitch* (pitch)

Set instrument transposition

**tuplet** [*music*] - *ratio* (fraction, as pair) *tuplet-span* [duration] *music* (music)

Scale the given *music* to tuplets. *ratio* is a fraction that specifies how many notes are played in place of the nominal value: it will be ‘3/2’ for triplets, namely three notes being played in place of two. If the optional duration *tuplet-span* is specified, it is used instead of **tupletSpannerDuration** for grouping the tuplets. For example, `\tuplet 3/2 4 { c8 c c c c c }`

will result in two groups of three tuplets, each group lasting for a quarter note.

**tupletSpan** [music] - *tuplet-span* [duration]

Set **tupletSpannerDuration**, the length into which **\tuplet** without an explicit ‘**tuplet-span**’ argument of its own will group its tuplets, to the duration *tuplet-span*. To revert to the default of not subdividing the contents of a **\tuplet** command without explicit ‘**tuplet-span**’, use

**\tupletSpan \default**

**tweak** [music] - *prop* (symbol list or symbol) *value* (any type) *music* (music)

Add a tweak to the following *music*. Layout objects created by *music* get their property *prop* set to *value*. If *prop* has the form ‘**Grob.property**’, like with

**\tweak Accidental.color #red cis'**

an indirectly created grob (‘**Accidental**’ is caused by ‘**NoteHead**’) can be tweaked; otherwise only directly created grobs are affected.

*prop* can contain additional elements in which case a nested property (inside of an alist) is tweaked.

**undo** [music] - *music* (music)

Convert **\override** and **\set** in *music* to **\revert** and **\unset**, respectively. Any reverts and unsets already in *music* cause a warning. Non-property-related music is ignored.

**unfoldRepeats** [music] - *music* (music)

Force any **\repeat volta**, **\repeat tremolo** or **\repeat percent** commands in *music* to be interpreted as **\repeat unfold**.

**void** [void] - *arg* (any type)

Accept a scheme argument, return a void expression. Use this if you want to have a scheme expression evaluated because of its side-effects, but its value ignored.

**withMusicProperty** [music] - *sym* (symbol) *val* (any type) *music* (music)

Set *sym* to *val* in *music*.

**xNote** [music] - *note* (music)

Print *note* with a cross-shaped note head.

**xNotesOn** [music]

Set the default note head style to a cross-shaped style.

**\=** [post event] - *id* (number or string) *event* (post event)

This sets the **spanner-id** property of the following *event* to the given *id* (numbers will be converted to a string). This can be used to tell LilyPond how to connect overlapping or parallel slurs or phrasing slurs within a single Voice.

**\fixed c' { c\=1( d\=2( e\=1) f\=2) }**



## A.20 Identificatori delle modifiche di contesto

I seguenti comandi possono essere usati come modificatori di contesto all’interno di un blocco **\layout** o **\with**.

**RemoveAllEmptyStaves**

Remove staves which are considered to be empty according to the list of interfaces set by **keepAliveInterfaces**, including those in the first system.



- Sets grob property `remove-empty` in Sezione ‘‘VerticalAxisGroup’’ in *Guida al Funzionamento Interno* to #t.
- Sets grob property `remove-first` in Sezione ‘‘VerticalAxisGroup’’ in *Guida al Funzionamento Interno* to #t.

#### RemoveEmptyStaves

Remove staves which are considered to be empty according to the list of interfaces set by `keepAliveInterfaces`.

- Sets grob property `remove-empty` in Sezione ‘‘VerticalAxisGroup’’ in *Guida al Funzionamento Interno* to #t.

## A.21 Tipi di predicati predefiniti

### R5RS primary predicates

Type predicate	Description
<code>boolean?</code>	boolean
<code>char?</code>	character
<code>number?</code>	number
<code>pair?</code>	pair
<code>port?</code>	port
<code>procedure?</code>	procedure
<code>string?</code>	string
<code>symbol?</code>	symbol
<code>vector?</code>	vector

### R5RS secondary predicates

Type predicate	Description
<code>char-alphabetic?</code>	alphabetic character
<code>char-lower-case?</code>	lower-case character
<code>char-numeric?</code>	numeric character
<code>char-upper-case?</code>	upper-case character
<code>char-whitespace?</code>	whitespace character
<code>complex?</code>	complex number
<code>eof-object?</code>	end-of-file object
<code>even?</code>	even number
<code>exact?</code>	exact number
<code>inexact?</code>	inexact number
<code>input-port?</code>	input port
<code>integer?</code>	integer
<code>list?</code>	list ( <i>use <code>cheap-list?</code> for faster processing</i> )
<code>negative?</code>	negative number
<code>null?</code>	null
<code>odd?</code>	odd number
<code>output-port?</code>	output port
<code>positive?</code>	positive number
<code>rational?</code>	rational number
<code>real?</code>	real number
<code>zero?</code>	zero

## Guile predicates

Type predicate	Description
hash-table?	hash table

## LilyPond scheme predicates

Type predicate	Description
boolean-or-symbol?	boolean or symbol
cheap-list?	list ( <i>use this instead of list? for faster processing</i> )
color?	color
fraction?	fraction, as pair
grob-list?	list of grobs
index?	non-negative integer
markup?	markup
markup-command-list?	markup command list
markup-list?	markup list
moment-pair?	pair of moment objects
number-list?	number list
number-or-grob?	number or grob
number-or-markup?	number or markup
number-or-pair?	number or pair
number-or-string?	number or string
number-pair?	pair of numbers
number-pair-list?	list of number pairs
rational-or-procedure?	an exact rational or procedure
rhythmic-location?	rhythmic location
scheme?	any type
string-or-music?	string or music
string-or-pair?	string or pair
string-or-symbol?	string or symbol
symbol-list?	symbol list
symbol-list-or-music?	symbol list or music
symbol-list-or-symbol?	symbol list or symbol
void?	void

## LilyPond exported predicates

Type predicate	Description
ly:book?	book
ly:box?	box
ly:context?	context
ly:context-def?	context definition
ly:context-mod?	context modification
ly:dimension?	dimension, in staff space
ly:dir?	direction
ly:dispatcher?	dispatcher
ly:duration?	duration
ly:event?	post event
ly:font-metric?	font metric
ly:grob?	graphical (layout) object

<code>ly:grob-array?</code>	array of grobs
<code>ly:grob-properties?</code>	grob properties
<code>ly:input-location?</code>	input location
<code>ly:item?</code>	item
<code>ly:iterator?</code>	iterator
<code>ly:lily-lexer?</code>	lily-lexer
<code>ly:lily-parser?</code>	lily-parser
<code>ly:listener?</code>	listener
<code>ly:moment?</code>	moment
<code>ly:music?</code>	music
<code>ly:music-function?</code>	music function
<code>ly:music-list?</code>	list of music objects
<code>ly:music-output?</code>	music output
<code>ly:otf-font?</code>	OpenType font
<code>ly:output-def?</code>	output definition
<code>ly:page-marker?</code>	page marker
<code>ly:pango-font?</code>	pango font
<code>ly:paper-book?</code>	paper book
<code>ly:paper-system?</code>	paper-system Prob
<code>ly:pitch?</code>	pitch
<code>ly:prob?</code>	property object
<code>ly:score?</code>	score
<code>ly:skyline?</code>	skyline
<code>ly:skyline-pair?</code>	pair of skylines
<code>ly:source-file?</code>	source file
<code>ly:spanner?</code>	spanner
<code>ly:spring?</code>	spring
<code>ly:stencil?</code>	stencil
<code>ly:stream-event?</code>	stream event
<code>ly:translator?</code>	translator
<code>ly:translator-group?</code>	translator group
<code>ly:undead?</code>	undead container
<code>ly:unpure-pure-container?</code>	unpure/pure container

## A.22 Funzioni Scheme

<code>ly:add-context-mod</code>	<i>contextmods</i>	<i>modification</i>	[Funzione]	
Adds the given context <i>modification</i> to the list <i>contextmods</i> of context modifications.				
<code>ly:add-file-name-alist</code>	<i>alist</i>		[Funzione]	
Add mappings for error messages from <i>alist</i> .				
<code>ly:add-interface</code>	<i>iface</i>	<i>desc</i>	<i>props</i>	[Funzione]
Add a new grob interface. <i>iface</i> is the interface name, <i>desc</i> is the interface description, and <i>props</i> is the list of user-settable properties for the interface.				
<code>ly:add-listener</code>	<i>callback</i>	<i>disp</i>	<i>cl</i>	[Funzione]
Add the single-argument procedure <i>callback</i> as listener to the dispatcher <i>disp</i> . Whenever <i>disp</i> hears an event of class <i>cl</i> , it calls <i>callback</i> with it.				
<code>ly:add-option</code>	<i>sym</i>	<i>val</i>	<i>description</i>	[Funzione]
Add a program option <i>sym</i> . <i>val</i> is the default value and <i>description</i> is a string description.				

<b>ly:all-grob-interfaces</b>	[Funzione]
Return the hash table with all grob interface descriptions.	
<b>ly:all-options</b>	[Funzione]
Get all option settings in an alist.	
<b>ly:all-stencil-expressions</b>	[Funzione]
Return all symbols recognized as stencil expressions.	
<b>ly:assoc-get</b> <i>key alist default-value strict-checking</i>	[Funzione]
Return value if <i>key</i> in <i>alist</i> , else <i>default-value</i> (or <b>#f</b> if not specified). If <i>strict-checking</i> is set to <b>#t</b> and <i>key</i> is not in <i>alist</i> , a <code>programming-error</code> is output.	
<b>ly:axis-group-interface::add-element</b> <i>grob grob-element</i>	[Funzione]
Set <i>grob</i> the parent of <i>grob-element</i> on all axes of <i>grob</i> .	
<b>ly:basic-progress</b> <i>str rest</i>	[Funzione]
A Scheme callable function to issue a basic progress message <i>str</i> . The message is formatted with <b>format</b> and <i>rest</i> .	
<b>ly:beam-score-count</b>	[Funzione]
count number of beam scores.	
<b>ly:bigpdfs</b>	[Funzione]
Return true if the command line includes the <b>--bigpdf</b> parameter.	
<b>ly:book?</b> <i>x</i>	[Funzione]
Is <i>x</i> a Book object?	
<b>ly:book-add-bookpart!</b> <i>book-smob book-part</i>	[Funzione]
Add <i>book-part</i> to <i>book-smob</i> book part list.	
<b>ly:book-add-score!</b> <i>book-smob score</i>	[Funzione]
Add <i>score</i> to <i>book-smob</i> score list.	
<b>ly:book-book-parts</b> <i>book</i>	[Funzione]
Return book parts in <i>book</i> .	
<b>ly:book-header</b> <i>book</i>	[Funzione]
Return header in <i>book</i> .	
<b>ly:book-paper</b> <i>book</i>	[Funzione]
Return paper in <i>book</i> .	
<b>ly:book-process</b> <i>book-smob default-paper default-layout output</i>	[Funzione]
Print book. <i>output</i> is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).	
<b>ly:book-process-to-systems</b> <i>book-smob default-paper default-layout output</i>	[Funzione]
Print book. <i>output</i> is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).	
<b>ly:book-scores</b> <i>book</i>	[Funzione]
Return scores in <i>book</i> .	
<b>ly:book-set-header!</b> <i>book module</i>	[Funzione]
Set the book header.	

<code>ly:box? x</code>	[Funzione]
Is <i>x</i> a <code>Box</code> object?	
<code>ly:bp num</code>	[Funzione]
<i>num</i> bigpoints (1/72th inch).	
<code>ly:bracket a iv t p</code>	[Funzione]
Make a bracket in direction <i>a</i> . The extent of the bracket is given by <i>iv</i> . The wings protrude by an amount of <i>p</i> , which may be negative. The thickness is given by <i>t</i> .	
<code>ly:broadcast disp ev</code>	[Funzione]
Send the stream event <i>ev</i> to the dispatcher <i>disp</i> .	
<code>ly:camel-case-&gt;lisp-identifier name-sym</code>	[Funzione]
Convert <code>FooBar_Bla</code> to <code>foo-bar-bla</code> style symbol.	
<code>ly:chain-assoc-get key achain default-value strict-checking</code>	[Funzione]
Return value for <i>key</i> from a list of alists <i>achain</i> . If no entry is found, return <i>default-value</i> or <code>#f</code> if <i>default-value</i> is not specified. With <i>strict-checking</i> set to <code>#t</code> , a <code>programming_error</code> is output in such cases.	
<code>ly:check-expected-warnings</code>	[Funzione]
Check whether all expected warnings have really been triggered.	
<code>ly:cm num</code>	[Funzione]
<i>num</i> cm.	
<code>ly:command-line-code</code>	[Funzione]
The Scheme code specified on command-line with <code>-e</code> .	
<code>ly:command-line-options</code>	[Funzione]
The Scheme options specified on command-line with <code>-d</code> .	
<code>ly:connect-dispatchers to from</code>	[Funzione]
Make the dispatcher <i>to</i> listen to events from <i>from</i> .	
<code>ly:context? x</code>	[Funzione]
Is <i>x</i> a <code>Context</code> object?	
<code>ly:context-current-moment context</code>	[Funzione]
Return the current moment of <i>context</i> .	
<code>ly:context-def? x</code>	[Funzione]
Is <i>x</i> a <code>Context_def</code> object?	
<code>ly:context-def-lookup def sym val</code>	[Funzione]
Return the value of <i>sym</i> in context definition <i>def</i> (e.g., <code>\Voice</code> ). If no value is found, return <i>val</i> or <code>()</code> if <i>val</i> is undefined. <i>sym</i> can be any of <code>'default-child'</code> , <code>'consists'</code> , <code>'description'</code> , <code>'aliases'</code> , <code>'accepts'</code> , <code>'property-ops'</code> , <code>'context-name'</code> , <code>'group-type'</code> .	
<code>ly:context-def-modify def mod</code>	[Funzione]
Return the result of applying the context-mod <i>mod</i> to the context definition <i>def</i> . Does not change <i>def</i> .	
<code>ly:context-event-source context</code>	[Funzione]
Return <code>event-source</code> of context <i>context</i> .	

<code>ly:context-events-below</code>	<i>context</i>	[Funzione]
Return a <code>stream-distributor</code> that distributes all events from <i>context</i> and all its subcontexts.		
<code>ly:context-find</code>	<i>context name</i>	[Funzione]
Find a parent of <i>context</i> that has name or alias <i>name</i> . Return <code>#f</code> if not found.		
<code>ly:context-grob-definition</code>	<i>context name</i>	[Funzione]
Return the definition of <i>name</i> (a symbol) within <i>context</i> as an alist.		
<code>ly:context-id</code>	<i>context</i>	[Funzione]
Return the ID string of <i>context</i> , i.e., for <code>\context Voice = "one" ...</code> return the string <code>one</code> .		
<code>ly:context-matched-pop-property</code>	<i>context grob cell</i>	[Funzione]
This undoes a particular <code>\override</code> , <code>\once \override</code> or <code>\once \revert</code> when given the specific alist pair to undo.		
<code>ly:context-mod?</code>	<i>x</i>	[Funzione]
Is <i>x</i> a <code>Context_mod</code> object?		
<code>ly:context-mod-apply!</code>	<i>context mod</i>	[Funzione]
Apply the context modification <i>mod</i> to <i>context</i> .		
<code>ly:context-name</code>	<i>context</i>	[Funzione]
Return the name of <i>context</i> , i.e., for <code>\context Voice = "one" ...</code> return the symbol <code>Voice</code> .		
<code>ly:context-now</code>	<i>context</i>	[Funzione]
Return <code>now-moment</code> of context <i>context</i> .		
<code>ly:context-parent</code>	<i>context</i>	[Funzione]
Return the parent of <i>context</i> , <code>#f</code> if none.		
<code>ly:context-property</code>	<i>context sym def</i>	[Funzione]
Return the value for property <i>sym</i> in <i>context</i> . If <i>def</i> is given, and property value is <code>'()</code> , return <i>def</i> .		
<code>ly:context-property-where-defined</code>	<i>context name</i>	[Funzione]
Return the context above <i>context</i> where <i>name</i> is defined.		
<code>ly:context-pushpop-property</code>	<i>context grob eltprop val</i>	[Funzione]
Do <code>\temporary \override</code> or <code>\revert</code> operation in <i>context</i> . The grob definition <i>grob</i> is extended with <i>eltprop</i> (if <i>val</i> is specified) or reverted (if unspecified).		
<code>ly:context-set-property!</code>	<i>context name val</i>	[Funzione]
Set value of property <i>name</i> in context <i>context</i> to <i>val</i> .		
<code>ly:context-unset-property</code>	<i>context name</i>	[Funzione]
Unset value of property <i>name</i> in context <i>context</i> .		
<code>ly:debug</code>	<i>str rest</i>	[Funzione]
A Scheme callable function to issue a debug message <i>str</i> . The message is formatted with <i>format</i> and <i>rest</i> .		
<code>ly:default-scale</code>		[Funzione]
Get the global default scale.		
<code>ly:dimension?</code>	<i>d</i>	[Funzione]
Return <i>d</i> as a number. Used to distinguish length variables from normal numbers.		

<code>ly:dir? s</code>	[Funzione]
Is <i>s</i> a direction? Valid directions are -1, 0, or 1, where -1 represents left or down, 1 represents right or up, and 0 represents a neutral direction.	
<code>ly:disconnect-dispatchers to from</code>	[Funzione]
Stop the dispatcher <i>to</i> listening to events from <i>from</i> .	
<code>ly:dispatcher? x</code>	[Funzione]
Is <i>x</i> a Dispatcher object?	
<code>ly:duration? x</code>	[Funzione]
Is <i>x</i> a Duration object?	
<code>ly:duration&lt;? p1 p2</code>	[Funzione]
Is <i>p1</i> shorter than <i>p2</i> ?	
<code>ly:duration-&gt;string dur</code>	[Funzione]
Convert <i>dur</i> to a string.	
<code>ly:duration-dot-count dur</code>	[Funzione]
Extract the dot count from <i>dur</i> .	
<code>ly:duration-factor dur</code>	[Funzione]
Extract the compression factor from <i>dur</i> . Return it as a pair.	
<code>ly:duration-length dur</code>	[Funzione]
The length of the duration as a <b>moment</b> .	
<code>ly:duration-log dur</code>	[Funzione]
Extract the duration log from <i>dur</i> .	
<code>ly:duration-scale dur</code>	[Funzione]
Extract the compression factor from <i>dur</i> . Return it as a rational.	
<code>ly:effective-prefix</code>	[Funzione]
Return effective prefix.	
<code>ly:encode-string-for-pdf str</code>	[Funzione]
Encode the given string to either Latin1 (which is a subset of the PDFDocEncoding) or if that's not possible to full UTF-16BE with Byte-Order-Mark (BOM).	
<code>ly:engraver-announce-end-grob engraver grob cause</code>	[Funzione]
Announce the end of a grob (i.e., the end of a spanner) originating from given <i>engraver</i> instance, with <i>grob</i> being a grob. <i>cause</i> should either be another grob or a music event.	
<code>ly:engraver-make-grob engraver grob-name cause</code>	[Funzione]
Create a grob originating from given <i>engraver</i> instance, with given <i>grob-name</i> , a symbol. <i>cause</i> should either be another grob or a music event.	
<code>ly:error str rest</code>	[Funzione]
A Scheme callable function to issue the error <i>str</i> . The error is formatted with <b>format</b> and <i>rest</i> .	
<code>ly:event? obj</code>	[Funzione]
Is <i>obj</i> a proper (non-rhythmic) event object?	
<code>ly:event-deep-copy m</code>	[Funzione]
Copy <i>m</i> and all sub expressions of <i>m</i> .	

- ly:event-property** *sev sym val* [Funzione]  
 Get the property *sym* of stream event *sev*. If *sym* is undefined, return *val* or '()' if *val* is not specified.
- ly:event-set-property!** *ev sym val* [Funzione]  
 Set property *sym* in event *ev* to *val*.
- ly:expand-environment** *str* [Funzione]  
 Expand *\$VAR* and *\${VAR}* in *str*.
- ly:expect-warning** *str rest* [Funzione]  
 A Scheme callable function to register a warning to be expected and subsequently suppressed. If the warning is not encountered, a warning about the missing warning will be shown. The message should be translated with (*\_ ...*) and changing parameters given after the format string.
- ly:find-file** *name* [Funzione]  
 Return the absolute file name of *name*, or *#f* if not found.
- ly:font-config-add-directory** *dir* [Funzione]  
 Add directory *dir* to FontConfig.
- ly:font-config-add-font** *font* [Funzione]  
 Add font *font* to FontConfig.
- ly:font-config-display-fonts** [Funzione]  
 Dump a list of all fonts visible to FontConfig.
- ly:font-config-get-font-file** *name* [Funzione]  
 Get the file for font *name*.
- ly:font-design-size** *font* [Funzione]  
 Given the font metric *font*, return the design size, relative to the current output-scale.
- ly:font-file-name** *font* [Funzione]  
 Given the font metric *font*, return the corresponding file name.
- ly:font-get-glyph** *font name* [Funzione]  
 Return a stencil from *font* for the glyph named *name*. If the glyph is not available, return an empty stencil.  
 Note that this command can only be used to access glyphs from fonts loaded with **ly:system-font-load**; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings **fetaMusic** and **fetaBraces**, respectively.
- ly:font-glyph-name-to-charset** *font name* [Funzione]  
 Return the character code for glyph *name* in *font*.  
 Note that this command can only be used to access glyphs from fonts loaded with **ly:system-font-load**; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings **fetaMusic** and **fetaBraces**, respectively.
- ly:font-glyph-name-to-index** *font name* [Funzione]  
 Return the index for *name* in *font*.  
 Note that this command can only be used to access glyphs from fonts loaded with **ly:system-font-load**; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings **fetaMusic** and **fetaBraces**, respectively.



- ly:font-index-to-charcode** *font index* [Funzione]  
 Return the character code for *index* in *font*.  
 Note that this command can only be used to access glyphs from fonts loaded with **ly:system-font-load**; currently, this means either the Emmentaler or Emmentaler-Brace fonts, corresponding to the font encodings **fetaMusic** and **fetaBraces**, respectively.
- ly:font-magnification** *font* [Funzione]  
 Given the font metric *font*, return the magnification, relative to the current output-scale.
- ly:font-metric?** *x* [Funzione]  
 Is *x* a **Font\_metric** object?
- ly:font-name** *font* [Funzione]  
 Given the font metric *font*, return the corresponding name.
- ly:font-sub-fonts** *font* [Funzione]  
 Given the font metric *font* of an OpenType font, return the names of the subfonts within *font*.
- ly:format** *str rest* [Funzione]  
 LilyPond specific format, supporting *~a* and *~[0-9]f*. Basic support for *~s* is also provided.
- ly:format-output** *context* [Funzione]  
 Given a global context in its final state, process it and return the **Music\_output** object in its final state.
- ly:generic-bound-extent** *grob common* [Funzione]  
 Determine the extent of *grob* relative to *common* along the X axis, finding its extent as a bound when it has **bound-alignment-interfaces** property list set and otherwise the full extent.
- ly:get-all-function-documentation** [Funzione]  
 Get a hash table with all LilyPond Scheme extension functions.
- ly:get-all-translators** [Funzione]  
 Return a list of all translator objects that may be instantiated.
- ly:get-context-mods** *contextmod* [Funzione]  
 Returns the list of context modifications stored in *contextmod*.
- ly:get-option** *var* [Funzione]  
 Get a global option setting.
- ly:get-spacing-spec** *from-scm to-scm* [Funzione]  
 Return the spacing spec going between the two given grobs, *from-scm* and *to-scm*.
- ly:get-undead** *undead* [Funzione]  
 Get back object from *undead*.
- ly:gettext** *original* [Funzione]  
 A Scheme wrapper function for **gettext**.
- ly:grob?** *x* [Funzione]  
 Is *x* a **Grob** object?
- ly:grob-alist-chain** *grob global* [Funzione]  
 Get an alist chain for grob *grob*, with *global* as the global default. If unspecified, **font-defaults** from the layout block is taken.

<code>ly:grob-array? x</code>	[Funzione]
Is <i>x</i> a <code>Grob_array</code> object?	
<code>ly:grob-array-&gt;list grob-arr</code>	[Funzione]
Return the elements of <i>grob-arr</i> as a Scheme list.	
<code>ly:grob-array-length grob-arr</code>	[Funzione]
Return the length of <i>grob-arr</i> .	
<code>ly:grob-array-ref grob-arr index</code>	[Funzione]
Retrieve the <i>index</i> th element of <i>grob-arr</i> .	
<code>ly:grob-basic-properties grob</code>	[Funzione]
Get the immutable properties of <i>grob</i> .	
<code>ly:grob-chain-callback grob proc sym</code>	[Funzione]
Find the callback that is stored as property <i>sym</i> of grob <i>grob</i> and chain <i>proc</i> to the head of this, meaning that it is called using <i>grob</i> and the previous callback's result.	
<code>ly:grob-common-refpoint grob other axis</code>	[Funzione]
Find the common refpoint of <i>grob</i> and <i>other</i> for <i>axis</i> .	
<code>ly:grob-common-refpoint-of-array grob others axis</code>	[Funzione]
Find the common refpoint of <i>grob</i> and <i>others</i> (a grob-array) for <i>axis</i> .	
<code>ly:grob-default-font grob</code>	[Funzione]
Return the default font for grob <i>grob</i> .	
<code>ly:grob-extent grob refp axis</code>	[Funzione]
Get the extent in <i>axis</i> direction of <i>grob</i> relative to the grob <i>refp</i> .	
<code>ly:grob-get-vertical-axis-group-index grob</code>	[Funzione]
Get the index of the vertical axis group the grob <i>grob</i> belongs to; return -1 if none is found.	
<code>ly:grob-interfaces grob</code>	[Funzione]
Return the interfaces list of grob <i>grob</i> .	
<code>ly:grob-layout grob</code>	[Funzione]
Get <code>\layout</code> definition from grob <i>grob</i> .	
<code>ly:grob-object grob sym</code>	[Funzione]
Return the value of a pointer in grob <i>grob</i> of property <i>sym</i> . It returns '() (end-of-list) if <i>sym</i> is undefined in <i>grob</i> .	
<code>ly:grob-original grob</code>	[Funzione]
Return the unbroken original grob of <i>grob</i> .	
<code>ly:grob-parent grob axis</code>	[Funzione]
Get the parent of <i>grob</i> . <i>axis</i> is 0 for the X-axis, 1 for the Y-axis.	
<code>ly:grob-pq&lt;? a b</code>	[Funzione]
Compare two grob priority queue entries. This is an internal function.	
<code>ly:grob-properties grob</code>	[Funzione]
Get the mutable properties of <i>grob</i> .	
<code>ly:grob-properties? x</code>	[Funzione]
Is <i>x</i> a <code>Grob_properties</code> object?	

- ly:grob-property** *grob sym val* [Funzione]  
Return the value for property *sym* of *grob*. If no value is found, return *val* or '()' if *val* is not specified.
- ly:grob-property-data** *grob sym* [Funzione]  
Return the value for property *sym* of *grob*, but do not process callbacks.
- ly:grob-pure-height** *grob refp beg end val* [Funzione]  
Return the pure height of *grob* given reffpoint *refp*. If no value is found, return *val* or '()' if *val* is not specified.
- ly:grob-pure-property** *grob sym beg end val* [Funzione]  
Return the pure value for property *sym* of *grob*. If no value is found, return *val* or '()' if *val* is not specified.
- ly:grob-relative-coordinate** *grob refp axis* [Funzione]  
Get the coordinate in *axis* direction of *grob* relative to the *grob refp*.
- ly:grob-robust-relative-extent** *grob refp axis* [Funzione]  
Get the extent in *axis* direction of *grob* relative to the *grob refp*, or (0,0) if empty.
- ly:grob-script-priority-less** *a b* [Funzione]  
Compare two grobs by script priority. For internal use.
- ly:grob-set-nested-property!** *grob symlist val* [Funzione]  
Set nested property *symlist* in *grob grob* to value *val*.
- ly:grob-set-object!** *grob sym val* [Funzione]  
Set *sym* in *grob grob* to value *val*.
- ly:grob-set-parent!** *grob axis parent-grob* [Funzione]  
Set *parent-grob* the parent of *grob grob* in axis *axis*.
- ly:grob-set-property!** *grob sym val* [Funzione]  
Set *sym* in *grob grob* to value *val*.
- ly:grob-spanned-rank-interval** *grob* [Funzione]  
Returns a pair with the **rank** of the furthest left column and the **rank** of the furthest right column spanned by *grob*.
- ly:grob-staff-position** *sg* [Funzione]  
Return the Y-position of *sg* relative to the staff.
- ly:grob-suicide!** *grob* [Funzione]  
Kill *grob*.
- ly:grob-system** *grob* [Funzione]  
Return the system *grob* of *grob*.
- ly:grob-translate-axis!** *grob d a* [Funzione]  
Translate *grob* on axis *a* over distance *d*.
- ly:grob-vertical<?** *a b* [Funzione]  
Does *a* lie above *b* on the page?
- ly:gulp-file** *name size* [Funzione]  
Read *size* characters from the file *name*, and return its contents in a string. If *size* is undefined, the entire file is read. The file is looked up using the search path.

<code>ly:hash-table-keys <i>tab</i></code>	[Funzione]
Return a list of keys in <i>tab</i> .	
<code>ly:inch <i>num</i></code>	[Funzione]
<i>num</i> inches.	
<code>ly:input-both-locations <i>sip</i></code>	[Funzione]
Return input location in <i>sip</i> as (file-name first-line first-column last-line last-column).	
<code>ly:input-file-line-char-column <i>sip</i></code>	[Funzione]
Return input location in <i>sip</i> as (file-name line char column).	
<code>ly:input-location? <i>x</i></code>	[Funzione]
Is <i>x</i> a Input object?	
<code>ly:input-message <i>sip msg rest</i></code>	[Funzione]
Print <i>msg</i> as a GNU compliant error message, pointing to the location in <i>sip</i> . <i>msg</i> is interpreted similar to <code>format</code> 's argument, using <i>rest</i> .	
<code>ly:input-warning <i>sip msg rest</i></code>	[Funzione]
Print <i>msg</i> as a GNU compliant warning message, pointing to the location in <i>sip</i> . <i>msg</i> is interpreted similar to <code>format</code> 's argument, using <i>rest</i> .	
<code>ly:interpret-music-expression <i>mus ctx</i></code>	[Funzione]
Interpret the music expression <i>mus</i> in the global context <i>ctx</i> . The context is returned in its final state.	
<code>ly:interpret-stencil-expression <i>expr func arg1 offset</i></code>	[Funzione]
Parse <i>expr</i> , feed bits to <i>func</i> with first arg <i>arg1</i> having offset <i>offset</i> .	
<code>ly:intlog2 <i>d</i></code>	[Funzione]
The 2-logarithm of 1/ <i>d</i> .	
<code>ly:item? <i>g</i></code>	[Funzione]
Is <i>g</i> an Item object?	
<code>ly:item-break-dir <i>it</i></code>	[Funzione]
The break status direction of item <i>it</i> . -1 means end of line, 0 unbroken, and 1 beginning of line.	
<code>ly:item-get-column <i>it</i></code>	[Funzione]
Return the PaperColumn or NonMusicalPaperColumn associated with this Item.	
<code>ly:iterator? <i>x</i></code>	[Funzione]
Is <i>x</i> a Music_iterator object?	
<code>ly:lexer-keywords <i>lexer</i></code>	[Funzione]
Return a list of (KEY . CODE) pairs, signifying the LilyPond reserved words list.	
<code>ly:lily-lexer? <i>x</i></code>	[Funzione]
Is <i>x</i> a Lily_lexer object?	
<code>ly:lily-parser? <i>x</i></code>	[Funzione]
Is <i>x</i> a Lily_parser object?	
<code>ly:line-interface::line <i>grob startx starty endx endy</i></code>	[Funzione]
Make a line using layout information from grob <i>grob</i> .	

- ly:listened-event-class?** *disp cl* [Funzione]  
Does *disp* listen to any event type in the list *cl*?
- ly:listened-event-types** *disp* [Funzione]  
Return a list of all event types that *disp* listens to.
- ly:listener?** *x* [Funzione]  
Is *x* a `Listener` object?
- ly:make-book** *paper header scores* [Funzione]  
Make a `\book` of *paper* and *header* (which may be `#f` as well) containing `\scores`.
- ly:make-book-part** *scores* [Funzione]  
Make a `\bookpart` containing `\scores`.
- ly:make-context-mod** *mod-list* [Funzione]  
Creates a context modification, optionally initialized via the list of modifications *mod-list*.
- ly:make-dispatcher** [Funzione]  
Return a newly created dispatcher.
- ly:make-duration** *length dotcount num den* [Funzione]  
*length* is the negative logarithm (base 2) of the duration: 1 is a half note, 2 is a quarter note, 3 is an eighth note, etc. The number of dots after the note is given by the optional argument *dotcount*.  
The duration factor is optionally given by integers *num* and *den*, alternatively by a single rational number.  
A duration is a musical duration, i.e., a length of time described by a power of two (whole, half, quarter, etc.) and a number of augmentation dots.
- ly:make-global-context** *output-def* [Funzione]  
Set up a global interpretation context, using the output block *output-def*. The context is returned.
- ly:make-global-translator** *global* [Funzione]  
Create a translator group and connect it to the global context *global*. The translator group is returned.
- ly:make-grob-properties** *alist* [Funzione]  
This packages the given property list *alist* in a grob property container stored in a context property with the name of a grob.
- ly:make-moment** *m g gn gd* [Funzione]  
Create the moment with rational main timing *m*, and optional grace timing *g*.  
A *moment* is a point in musical time. It consists of a pair of rationals (*m*, *g*), where *m* is the timing for the main notes, and *g* the timing for grace notes. In absence of grace notes, *g* is zero.  
For compatibility reasons, it is possible to write two numbers specifying numerator and denominator instead of the rationals. These forms cannot be mixed, and the two-argument form is disambiguated by the sign of the second argument: if it is positive, it can only be a denominator and not a grace timing.
- ly:make-music** *props* [Funzione]  
Make a C++ `Music` object and initialize it with *props*.  
This function is for internal use and is only called by `make-music`, which is the preferred interface for creating music objects.

- ly:make-music-function** *signature func* [Funzione]  
 Make a function to process music, to be used for the parser. *func* is the function, and *signature* describes its arguments. *signature*'s cdr is a list containing either **ly:music?** predicates or other type predicates. Its car is the syntax function to call.
- ly:make-music-relative!** *music pitch* [Funzione]  
 Make *music* relative to *pitch*, return final pitch.
- ly:make-output-def** [Funzione]  
 Make an output definition.
- ly:make-page-label-marker** *label* [Funzione]  
 Return page marker with label *label*.
- ly:make-page-permission-marker** *symbol permission* [Funzione]  
 Return page marker with page breaking and turning permissions.
- ly:make-pango-description-string** *chain size* [Funzione]  
 Make a PangoFontDescription string for the property alist *chain* at size *size*.
- ly:make-paper-outputter** *port format* [Funzione]  
 Create an outputter that evaluates within **output-format**, writing to *port*.
- ly:make-pitch** *octave note alter* [Funzione]  
*octave* is specified by an integer, zero for the octave containing middle C. *note* is a number indexing the global default scale, with 0 corresponding to pitch C and 6 usually corresponding to pitch B. Optional *alter* is a rational number of 200-cent whole tones for alteration.
- ly:make-prob** *type init rest* [Funzione]  
 Create a Prob object.
- ly:make-scale** *steps* [Funzione]  
 Create a scale. The argument is a vector of rational numbers, each of which represents the number of 200 cent tones of a pitch above the tonic.
- ly:make-score** *music* [Funzione]  
 Return score with *music* encapsulated in it.
- ly:make-spring** *ideal min-dist* [Funzione]  
 Make a spring. *ideal* is the ideal distance of the spring, and *min-dist* is the minimum distance.
- ly:make-stencil** *expr xext yext* [Funzione]  
 Stencils are device independent output expressions. They carry two pieces of information:
1. A specification of how to print this object. This specification is processed by the output backends, for example **scm/output-ps.scm**.
  2. The vertical and horizontal extents of the object, given as pairs. If an extent is unspecified (or if you use **empty-interval** as its value), it is taken to be empty.
- ly:make-stream-event** *cl proplist* [Funzione]  
 Create a stream event of class *cl* with the given mutable property list.
- ly:make-undead** *object* [Funzione]  
 This packages *object* in a manner that keeps it from triggering "Parsed object should be dead" messages.

<code>ly:make-unpure-pure-container</code>	<i>unpure pure</i>	[Funzione]
Make an unpure-pure container. <i>unpure</i> should be an unpure expression, and <i>pure</i> should be a pure expression. If <i>pure</i> is omitted, the value of <i>unpure</i> will be used twice, except that a callback is given two extra arguments that are ignored for the sake of pure calculations.		
<code>ly:message</code>	<i>str rest</i>	[Funzione]
A Scheme callable function to issue the message <i>str</i> . The message is formatted with <b>format</b> and <i>rest</i> .		
<code>ly:minimal-breaking</code>	<i>pb</i>	[Funzione]
Break (pages and lines) the <b>Paper_book</b> object <i>pb</i> without looking for optimal spacing: stack as many lines on a page before moving to the next one.		
<code>ly:mm</code>	<i>num</i>	[Funzione]
<i>num</i> mm.		
<code>ly:module-&gt;alist</code>	<i>mod</i>	[Funzione]
Dump the contents of module <i>mod</i> as an alist.		
<code>ly:module-copy</code>	<i>dest src</i>	[Funzione]
Copy all bindings from module <i>src</i> into <i>dest</i> .		
<code>ly:modules-lookup</code>	<i>modules sym def</i>	[Funzione]
Look up <i>sym</i> in the list <i>modules</i> , returning the first occurrence. If not found, return <i>def</i> or <b>#f</b> if <i>def</i> isn't specified.		
<code>ly:moment?</code>	<i>x</i>	[Funzione]
Is <i>x</i> a <b>Moment</b> object?		
<code>ly:moment&lt;?</code>	<i>a b</i>	[Funzione]
Compare two moments.		
<code>ly:moment-add</code>	<i>a b</i>	[Funzione]
Add two moments.		
<code>ly:moment-div</code>	<i>a b</i>	[Funzione]
Divide two moments.		
<code>ly:moment-grace</code>	<i>mom</i>	[Funzione]
Extract grace timing as a rational number from <i>mom</i> .		
<code>ly:moment-grace-denominator</code>	<i>mom</i>	[Funzione]
Extract denominator from grace timing.		
<code>ly:moment-grace-numerator</code>	<i>mom</i>	[Funzione]
Extract numerator from grace timing.		
<code>ly:moment-main</code>	<i>mom</i>	[Funzione]
Extract main timing as a rational number from <i>mom</i> .		
<code>ly:moment-main-denominator</code>	<i>mom</i>	[Funzione]
Extract denominator from main timing.		
<code>ly:moment-main-numerator</code>	<i>mom</i>	[Funzione]
Extract numerator from main timing.		
<code>ly:moment-mod</code>	<i>a b</i>	[Funzione]
Modulo of two moments.		

<code>ly:moment-mul a b</code>	[Funzione]
Multiply two moments.	
<code>ly:moment-sub a b</code>	[Funzione]
Subtract two moments.	
<code>ly:music? obj</code>	[Funzione]
Is <i>obj</i> a music object?	
<code>ly:music-compress m factor</code>	[Funzione]
Compress music object <i>m</i> by moment <i>factor</i> .	
<code>ly:music-deep-copy m origin</code>	[Funzione]
Copy <i>m</i> and all sub expressions of <i>m</i> . <i>m</i> may be an arbitrary type; cons cells and music are copied recursively. If <i>origin</i> is given, it is used as the origin for one level of music by calling <code>ly:set-origin!</code> on the copy.	
<code>ly:music-duration-compress mus fact</code>	[Funzione]
Compress <i>mus</i> by factor <i>fact</i> , which is a <b>Moment</b> .	
<code>ly:music-duration-length mus</code>	[Funzione]
Extract the duration field from <i>mus</i> and return the length.	
<code>ly:music-function? x</code>	[Funzione]
Is <i>x</i> a <b>Music_function</b> object?	
<code>ly:music-function-extract x</code>	[Funzione]
Return the Scheme function inside <i>x</i> .	
<code>ly:music-function-signature x</code>	[Funzione]
Return the function signature inside <i>x</i> .	
<code>ly:music-length mus</code>	[Funzione]
Get the length of music expression <i>mus</i> and return it as a <b>Moment</b> object.	
<code>ly:music-list? lst</code>	[Funzione]
Is <i>lst</i> a list of music objects?	
<code>ly:music-mutable-properties mus</code>	[Funzione]
Return an alist containing the mutable properties of <i>mus</i> . The immutable properties are not available, since they are constant and initialized by the <b>make-music</b> function.	
<code>ly:music-output? x</code>	[Funzione]
Is <i>x</i> a <b>Music_output</b> object?	
<code>ly:music-property mus sym val</code>	[Funzione]
Return the value for property <i>sym</i> of music expression <i>mus</i> . If no value is found, return <i>val</i> or '() if <i>val</i> is not specified.	
<code>ly:music-set-property! mus sym val</code>	[Funzione]
Set property <i>sym</i> in music expression <i>mus</i> to <i>val</i> .	
<code>ly:music-transpose m p</code>	[Funzione]
Transpose <i>m</i> such that central C is mapped to <i>p</i> . Return <i>m</i> .	
<code>ly:note-column-accidentals note-column</code>	[Funzione]
Return the <b>AccidentalPlacement</b> grob from <i>note-column</i> if any, or <b>SCM_EOL</b> otherwise.	



<code>ly:note-column-dot-column</code> <i>note-column</i>	[Funzione]
Return the DotColumn grob from <i>note-column</i> if any, or SCM_EOL otherwise.	
<code>ly:note-head::stem-attachment</code> <i>font-metric</i> <i>glyph-name</i>	[Funzione]
Get attachment in <i>font-metric</i> for attaching a stem to notehead <i>glyph-name</i> .	
<code>ly:number-&gt;string</code> <i>s</i>	[Funzione]
Convert <i>s</i> to a string without generating many decimals.	
<code>ly:one-line-auto-height-breaking</code> <i>pb</i>	[Funzione]
Put each score on a single line, and put each line on its own page. Modify the paper-width setting so that every page is wider than the widest line. Modify the paper-height setting to fit the height of the tallest line.	
<code>ly:one-line-breaking</code> <i>pb</i>	[Funzione]
Put each score on a single line, and put each line on its own page. Modify the paper-width setting so that every page is wider than the widest line.	
<code>ly:optimal-breaking</code> <i>pb</i>	[Funzione]
Optimally break (pages and lines) the Paper_book object <i>pb</i> to minimize badness in both vertical and horizontal spacing.	
<code>ly:option-usage</code> <i>port</i>	[Funzione]
Print <code>ly:set-option</code> usage. Optional <i>port</i> argument for the destination defaults to current output port.	
<code>ly:otf-&gt;cff</code> <i>otf-file-name</i>	[Funzione]
Convert the contents of an OTF file to a CFF file, returning it as a string.	
<code>ly:otf-font?</code> <i>font</i>	[Funzione]
Is <i>font</i> an OpenType font?	
<code>ly:otf-font-glyph-info</code> <i>font</i> <i>glyph</i>	[Funzione]
Given the font metric <i>font</i> of an OpenType font, return the information about named glyph <i>glyph</i> (a string).	
<code>ly:otf-font-table-data</code> <i>font</i> <i>tag</i>	[Funzione]
Extract a table <i>tag</i> from <i>font</i> . Return empty string for non-existent <i>tag</i> .	
<code>ly:otf-glyph-count</code> <i>font</i>	[Funzione]
Return the number of glyphs in <i>font</i> .	
<code>ly:otf-glyph-list</code> <i>font</i>	[Funzione]
Return a list of glyph names for <i>font</i> .	
<code>ly:output-def?</code> <i>x</i>	[Funzione]
Is <i>x</i> a Output_def object?	
<code>ly:output-def-clone</code> <i>def</i>	[Funzione]
Clone output definition <i>def</i> .	
<code>ly:output-def-lookup</code> <i>def</i> <i>sym</i> <i>val</i>	[Funzione]
Return the value of <i>sym</i> in output definition <i>def</i> (e.g., <code>\paper</code> ). If no value is found, return <i>val</i> or '()' if <i>val</i> is undefined.	
<code>ly:output-def-parent</code> <i>def</i>	[Funzione]
Return the parent output definition of <i>def</i> .	

<code>ly:output-def-scope</code> <i>def</i>	[Funzione]
Return the variable scope inside <i>def</i> .	
<code>ly:output-def-set-variable!</code> <i>def sym val</i>	[Funzione]
Set an output definition <i>def</i> variable <i>sym</i> to <i>val</i> .	
<code>ly:output-description</code> <i>output-def</i>	[Funzione]
Return the description of translators in <i>output-def</i> .	
<code>ly:output-find-context-def</code> <i>output-def context-name</i>	[Funzione]
Return an alist of all context defs (matching <i>context-name</i> if given) in <i>output-def</i> .	
<code>ly:output-formats</code>	[Funzione]
Formats passed to <code>--format</code> as a list of strings, used for the output.	
<code>ly:outputter-close</code> <i>outputter</i>	[Funzione]
Close port of <i>outputter</i> .	
<code>ly:outputter-dump-stencil</code> <i>outputter stencil</i>	[Funzione]
Dump stencil <i>expr</i> onto <i>outputter</i> .	
<code>ly:outputter-dump-string</code> <i>outputter str</i>	[Funzione]
Dump <i>str</i> onto <i>outputter</i> .	
<code>ly:outputter-module</code> <i>outputter</i>	[Funzione]
Return output module of <i>outputter</i> .	
<code>ly:outputter-output-scheme</code> <i>outputter expr</i>	[Funzione]
Eval <i>expr</i> in module of <i>outputter</i> .	
<code>ly:outputter-port</code> <i>outputter</i>	[Funzione]
Return output port for <i>outputter</i> .	
<code>ly:page-marker?</code> <i>x</i>	[Funzione]
Is <i>x</i> a <code>Page_marker</code> object?	
<code>ly:page-turn-breaking</code> <i>pb</i>	[Funzione]
Optimally break (pages and lines) the <code>Paper_book</code> object <i>pb</i> such that page turns only happen in specified places, returning its pages.	
<code>ly:pango-font?</code> <i>f</i>	[Funzione]
Is <i>f</i> a pango font?	
<code>ly:pango-font-physical-fonts</code> <i>f</i>	[Funzione]
Return alist of ( <code>ps-name file-name font-index</code> ) lists for Pango font <i>f</i> .	
<code>ly:paper-book?</code> <i>x</i>	[Funzione]
Is <i>x</i> a <code>Paper_book</code> object?	
<code>ly:paper-book-header</code> <i>pb</i>	[Funzione]
Return the header definition ( <code>\header</code> ) in <code>Paper_book</code> object <i>pb</i> .	
<code>ly:paper-book-pages</code> <i>pb</i>	[Funzione]
Return pages in <code>Paper_book</code> object <i>pb</i> .	
<code>ly:paper-book-paper</code> <i>pb</i>	[Funzione]
Return the paper output definition ( <code>\paper</code> ) in <code>Paper_book</code> object <i>pb</i> .	

<code>ly:paper-book-performances</code> <i>pb</i>	[Funzione]
Return performances in <code>Paper_book</code> object <i>pb</i> .	
<code>ly:paper-book-scopes</code> <i>pb</i>	[Funzione]
Return scopes in <code>Paper_book</code> object <i>pb</i> .	
<code>ly:paper-book-systems</code> <i>pb</i>	[Funzione]
Return systems in <code>Paper_book</code> object <i>pb</i> .	
<code>ly:paper-column::print</code>	[Funzione]
Optional stencil for <code>PaperColumn</code> or <code>NonMusicalPaperColumn</code> . Draws the <b>rank number</b> of each column, its moment in time, a blue arrow showing the ideal distance, and a red arrow showing the minimum distance between columns.	
<code>ly:paper-fonts</code> <i>def</i>	[Funzione]
Return a list containing the fonts from output definition <i>def</i> (e.g., <code>\paper</code> ).	
<code>ly:paper-get-font</code> <i>def chain</i>	[Funzione]
Find a font metric in output definition <i>def</i> satisfying the font-qualifiers in alist chain <i>chain</i> , and return it. (An alist chain is a list of alists, containing grob properties.)	
<code>ly:paper-get-number</code> <i>def sym</i>	[Funzione]
Return the value of variable <i>sym</i> in output definition <i>def</i> as a double.	
<code>ly:paper-outputscales</code> <i>def</i>	[Funzione]
Return the output-scale for output definition <i>def</i> .	
<code>ly:paper-score-paper-systems</code> <i>paper-score</i>	[Funzione]
Return vector of <code>paper_system</code> objects from <i>paper-score</i> .	
<code>ly:paper-system?</code> <i>obj</i>	[Funzione]
Is <i>obj</i> a C++ Prob object of type <code>paper-system</code> ?	
<code>ly:paper-system-minimum-distance</code> <i>sys1 sys2</i>	[Funzione]
Measure the minimum distance between these two paper-systems, using their stored skylines if possible and falling back to their extents otherwise.	
<code>ly:parse-file</code> <i>name</i>	[Funzione]
Parse a single .ly file. Upon failure, throw <code>ly-file-failed</code> key.	
<code>ly:parse-string-expression</code> <i>parser-smob ly-code filename line</i>	[Funzione]
Parse the string <i>ly-code</i> with <i>parser-smob</i> . Return the contained music expression. <i>filename</i> and <i>line</i> are optional source indicators.	
<code>ly:parsed-undead-list!</code>	[Funzione]
Return the list of objects that have been found live that should have been dead, and clear that list.	
<code>ly:parser-clear-error</code> <i>parser</i>	[Funzione]
Clear error flag for <i>parser</i> , defaulting to current parser.	
<code>ly:parser-clone</code> <i>closures location</i>	[Funzione]
Return a clone of current parser. An association list of port positions to closures can be specified in <i>closures</i> in order to have <code>\$</code> and <code>#</code> interpreted in their original lexical environment. If <i>location</i> is a valid location, it becomes the source of all music expressions inside.	
<code>ly:parser-define!</code> <i>symbol val</i>	[Funzione]
Bind <i>symbol</i> to <i>val</i> in current parser's module.	

<b>ly:parser-error</b> <i>msg input</i>	[Funzione]
Display an error message and make current parser fail. Without a current parser, trigger an ordinary error.	
<b>ly:parser-has-error?</b> <i>parser</i>	[Funzione]
Does <i>parser</i> (defaulting to current parser) have an error flag?	
<b>ly:parser-include-string</b> <i>ly-code</i>	[Funzione]
Include the string <i>ly-code</i> into the input stream for current parser. Can only be used in immediate Scheme expressions (\$ instead of #).	
<b>ly:parser-lexer</b> <i>parser</i>	[Funzione]
Return the lexer for <i>parser</i> , defaulting to current parser	
<b>ly:parser-lookup</b> <i>symbol</i>	[Funzione]
Look up <i>symbol</i> in current parser's module. Return '() if not defined.	
<b>ly:parser-output-name</b> <i>parser</i>	[Funzione]
Return the base name of the output file. If <i>parser</i> is left off, use currently active parser.	
<b>ly:parser-parse-string</b> <i>parser-smob ly-code</i>	[Funzione]
Parse the string <i>ly-code</i> with <i>parser-smob</i> . Upon failure, throw <b>ly-file-failed</b> key.	
<b>ly:parser-set-note-names</b> <i>names</i>	[Funzione]
Replace current note names in parser. <i>names</i> is an alist of symbols. This only has effect if the current mode is notes.	
<b>ly:performance-header</b> <i>performance</i>	[Funzione]
Return header of performance.	
<b>ly:performance-set-header!</b> <i>performance module</i>	[Funzione]
Set the performance header.	
<b>ly:performance-write</b> <i>performance filename name</i>	[Funzione]
Write <i>performance</i> to <i>filename</i> storing <i>name</i> as the name of the performance in the file metadata.	
<b>ly:pfb-&gt;pfa</b> <i>pfb-file-name</i>	[Funzione]
Convert the contents of a Type 1 font in PFB format to PFA format.	
<b>ly:pitch?</b> <i>x</i>	[Funzione]
Is <i>x</i> a Pitch object?	
<b>ly:pitch&lt;?</b> <i>p1 p2</i>	[Funzione]
Is <i>p1</i> lexicographically smaller than <i>p2</i> ?	
<b>ly:pitch-alteration</b> <i>pp</i>	[Funzione]
Extract the alteration from pitch <i>pp</i> .	
<b>ly:pitch-diff</b> <i>pitch root</i>	[Funzione]
Return pitch <i>delta</i> such that <i>root</i> transposed by <i>delta</i> equals <i>pitch</i> .	
<b>ly:pitch-negate</b> <i>p</i>	[Funzione]
Negate <i>p</i> .	
<b>ly:pitch-notename</b> <i>pp</i>	[Funzione]
Extract the note name from pitch <i>pp</i> .	

<code>ly:pitch-octave pp</code>	[Funzione]
Extract the octave from pitch <i>pp</i> .	
<code>ly:pitch-quartertones pp</code>	[Funzione]
Calculate the number of quarter tones of <i>pp</i> from middle C.	
<code>ly:pitch-semitones pp</code>	[Funzione]
Calculate the number of semitones of <i>pp</i> from middle C.	
<code>ly:pitch-steps p</code>	[Funzione]
Number of steps counted from middle C of the pitch <i>p</i> .	
<code>ly:pitch-tones pp</code>	[Funzione]
Calculate the number of tones of <i>pp</i> from middle C as a rational number.	
<code>ly:pitch-transpose p delta</code>	[Funzione]
Transpose <i>p</i> by the amount <i>delta</i> , where <i>delta</i> is relative to middle C.	
<code>ly:pointer-group-interface::add-grob grob sym grob-element</code>	[Funzione]
Add <i>grob-element</i> to <i>grob</i> 's <i>sym</i> grob array.	
<code>ly:position-on-line? sg spos</code>	[Funzione]
Return whether <i>spos</i> is on a line of the staff associated with the grob <i>sg</i> (even on an extender line).	
<code>ly:prob? x</code>	[Funzione]
Is <i>x</i> a Prob object?	
<code>ly:prob-immutable-properties prob</code>	[Funzione]
Retrieve an alist of immutable properties.	
<code>ly:prob-mutable-properties prob</code>	[Funzione]
Retrieve an alist of mutable properties.	
<code>ly:prob-property prob sym val</code>	[Funzione]
Return the value for property <i>sym</i> of Prob object <i>prob</i> . If no value is found, return <i>val</i> or '()' if <i>val</i> is not specified.	
<code>ly:prob-property? obj sym</code>	[Funzione]
Is boolean prop <i>sym</i> of <i>obj</i> set?	
<code>ly:prob-set-property! obj sym value</code>	[Funzione]
Set property <i>sym</i> of <i>obj</i> to <i>value</i> .	
<code>ly:prob-type? obj type</code>	[Funzione]
Is <i>obj</i> the specified prob-type?	
<code>ly:programming-error str rest</code>	[Funzione]
A Scheme callable function to issue the internal warning <i>str</i> . The message is formatted with <b>format</b> and <i>rest</i> .	
<code>ly:progress str rest</code>	[Funzione]
A Scheme callable function to print progress <i>str</i> . The message is formatted with <b>format</b> and <i>rest</i> .	
<code>ly:property-lookup-stats sym</code>	[Funzione]
Return hash table with a property access corresponding to <i>sym</i> . Choices are <b>prob</b> , <b>grob</b> , and <b>context</b> .	

<b>ly:protects</b>	[Funzione]
Return hash of protected objects.	
<b>ly:pt <i>num</i></b>	[Funzione]
<i>num</i> printer points.	
<b>ly:pure-call <i>data grob start end rest</i></b>	[Funzione]
Convert property <i>data</i> (unpure-pure container or procedure) to value in a pure context defined by <i>grob</i> , <i>start</i> , <i>end</i> , and possibly <i>rest</i> arguments.	
<b>ly:register-stencil-expression <i>symbol</i></b>	[Funzione]
Add <i>symbol</i> as head of a stencil expression.	
<b>ly:relative-group-extent <i>elements common axis</i></b>	[Funzione]
Determine the extent of <i>elements</i> relative to <i>common</i> in the <i>axis</i> direction.	
<b>ly:reset-all-fonts</b>	[Funzione]
Forget all about previously loaded fonts.	
<b>ly:round-filled-box <i>xext yext blot</i></b>	[Funzione]
Make a <b>Stencil</b> object that prints a black box of dimensions <i>xext</i> , <i>yext</i> and roundness <i>blot</i> .	
<b>ly:round-filled-polygon <i>points blot extroversion</i></b>	[Funzione]
Make a <b>Stencil</b> object that prints a black polygon with corners at the points defined by <i>points</i> (list of coordinate pairs) and roundness <i>blot</i> . Optional <i>extroversion</i> shifts the outline outward, with the default of -1.0 keeping the outer boundary of the outline just inside of the polygon.	
<b>ly:run-translator <i>mus output-def</i></b>	[Funzione]
Process <i>mus</i> according to <i>output-def</i> . An interpretation context is set up, and <i>mus</i> is interpreted with it. The context is returned in its final state.	
Optionally, this routine takes an object-key to uniquely identify the score block containing it.	
<b>ly:score? <i>x</i></b>	[Funzione]
Is <i>x</i> a <b>Score</b> object?	
<b>ly:score-add-output-def! <i>score def</i></b>	[Funzione]
Add an output definition <i>def</i> to <i>score</i> .	
<b>ly:score-embedded-format <i>score layout</i></b>	[Funzione]
Run <i>score</i> through <i>layout</i> (an output definition) scaled to correct output-scale already, returning a list of layout-lines.	
<b>ly:score-error? <i>score</i></b>	[Funzione]
Was there an error in the score?	
<b>ly:score-header <i>score</i></b>	[Funzione]
Return score header.	
<b>ly:score-music <i>score</i></b>	[Funzione]
Return score music.	
<b>ly:score-output-defs <i>score</i></b>	[Funzione]
All output definitions in a score.	
<b>ly:score-set-header! <i>score module</i></b>	[Funzione]
Set the score header.	

- ly:separation-item::print** [Funzione]  
 Optional stencil for `PaperColumn` or `NonMusicalPaperColumn`. Draws the horizontal-skylines of each `PaperColumn`, showing the shapes used to determine the minimum distances between `PaperColumns` at the note-spacing step, before staves have been spaced (vertically) on the page.
- ly:set-default-scale** *scale* [Funzione]  
 Set the global default scale. This determines the tuning of pitches with no accidentals or key signatures. The first pitch is C. Alterations are calculated relative to this scale. The number of pitches in this scale determines the number of scale steps that make up an octave. Usually the 7-note major scale.
- ly:set-grob-modification-callback** *cb* [Funzione]  
 Specify a procedure that will be called every time LilyPond modifies a grob property. The callback will receive as arguments the grob that is being modified, the name of the C++ file in which the modification was requested, the line number in the C++ file in which the modification was requested, the name of the function in which the modification was requested, the property to be changed, and the new value for the property.
- ly:set-middle-C!** *context* [Funzione]  
 Set the `middleCPosition` variable in *context* based on the variables `middleCClefPosition` and `middleCOffset`.
- ly:set-option** *var val* [Funzione]  
 Set a program option.
- ly:set-origin!** *m origin* [Funzione]  
 This sets the origin given in *origin* to *m*. *m* will typically be a music expression or a list of music. List structures are searched recursively, but recursion stops at the changed music expressions themselves. *origin* is generally of type `ly:input-location?`, defaulting to `(*location*)`. Other valid values for *origin* are a music expression which is then used as the source of location information, or `#f` or `'()` in which case no action is performed. The return value is *m* itself.
- ly:set-property-cache-callback** *cb* [Funzione]  
 Specify a procedure that will be called whenever lilypond calculates a callback function and caches the result. The callback will receive as arguments the grob whose property it is, the name of the property, the name of the callback that calculated the property, and the new (cached) value of the property.
- ly:skyline?** *x* [Funzione]  
 Is *x* a `Skyline` object?
- ly:skyline-empty?** *sky* [Funzione]  
 Return whether *sky* is empty.
- ly:skyline-pair?** *x* [Funzione]  
 Is *x* a `Skyline_pair` object?
- ly:slur-score-count** [Funzione]  
 count number of slur scores.
- ly:smob-protects** [Funzione]  
 Return LilyPond's internal smob protection list.

- ly:solve-spring-rod-problem** *springs rods length ragged* [Funzione]  
 Solve a spring and rod problem for *count* objects, that are connected by *count-1* *springs*, and an arbitrary number of *rods*. *count* is implicitly given by *springs* and *rods*. The *springs* argument has the format (*ideal, inverse\_hook*) and *rods* is of the form (*idx1, idx2, distance*).  
*length* is a number, *ragged* a boolean.  
 The function returns a list containing the force (positive for stretching, negative for compressing and **#f** for non-satisfied constraints) followed by *spring-count+1* positions of the objects.
- ly:source-file?** *x* [Funzione]  
 Is *x* a **Source\_file** object?
- ly:spanner?** *g* [Funzione]  
 Is *g* a spanner object?
- ly:spanner-bound** *spanner dir* [Funzione]  
 Get one of the bounds of *spanner*. *dir* is **-1** for left, and **1** for right.
- ly:spanner-broken-into** *spanner* [Funzione]  
 Return broken-into list for *spanner*.
- ly:spanner-set-bound!** *spanner dir item* [Funzione]  
 Set grob *item* as bound in direction *dir* for *spanner*.
- ly:spawn** *command rest* [Funzione]  
 Simple interface to `g_spawn_sync` *str*. The error is formatted with **format** and *rest*.
- ly:spring?** *x* [Funzione]  
 Is *x* a **Spring** object?
- ly:spring-set-inverse-compress-strength!** *spring strength* [Funzione]  
 Set the inverse compress *strength* of *spring*.
- ly:spring-set-inverse-stretch-strength!** *spring strength* [Funzione]  
 Set the inverse stretch *strength* of *spring*.
- ly:staff-symbol-line-thickness** *grob* [Funzione]  
 Returns the current staff-line thickness in the staff associated with *grob*, expressed as a multiple of the current staff-space height.
- ly:staff-symbol-staff-radius** *grob* [Funzione]  
 Returns the radius of the staff associated with *grob*.
- ly:staff-symbol-staff-space** *grob* [Funzione]  
 Returns the current staff-space height in the staff associated with *grob*, expressed as a multiple of the default height of a staff-space in the traditional five-line staff.
- ly:start-environment** [Funzione]  
 Return the environment (a list of strings) that was in effect at program start.
- ly:stderr-redirect** *file-name mode* [Funzione]  
 Redirect stderr to *file-name*, opened with *mode*.
- ly:stencil?** *x* [Funzione]  
 Is *x* a **Stencil** object?









- ly:stencil-add** *args* [Funzione]  
Combine stencils. Takes any number of arguments.
- ly:stencil-aligned-to** *stil axis dir* [Funzione]  
Align *stil* using its own extents. *dir* is a number. -1 and 1 are left and right, respectively. Other values are interpolated (so 0 means the center).
- ly:stencil-combine-at-edge** *first axis direction second padding* [Funzione]  
Construct a stencil by putting *second* next to *first*. *axis* can be 0 (x-axis) or 1 (y-axis). *direction* can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with *padding* as extra space. *first* and *second* may also be '()' or #f.
- ly:stencil-empty?** *stil axis* [Funzione]  
Return whether *stil* is empty. If an optional *axis* is supplied, the emptiness check is restricted to that axis.
- ly:stencil-expr** *stil* [Funzione]  
Return the expression of *stil*.
- ly:stencil-extent** *stil axis* [Funzione]  
Return a pair of numbers signifying the extent of *stil* in *axis* direction (0 or 1 for x and y axis, respectively).
- ly:stencil-fonts** *s* [Funzione]  
Analyze *s*, and return a list of fonts used in *s*.
- ly:stencil-in-color** *stc r g b* [Funzione]  
Put *stc* in a different color.
- ly:stencil-rotate** *stil angle x y* [Funzione]  
Return a stencil *stil* rotated *angle* degrees around the relative offset (x, y). E.g., an offset of (-1, 1) will rotate the stencil around the left upper corner.
- ly:stencil-rotate-absolute** *stil angle x y* [Funzione]  
Return a stencil *stil* rotated *angle* degrees around point (x, y), given in absolute coordinates.
- ly:stencil-scale** *stil x y* [Funzione]  
Scale stencil *stil* using the horizontal and vertical scaling factors *x* and *y*. Negative values will flip or mirror *stil* without changing its origin; this may result in collisions unless it is repositioned.
- ly:stencil-stack** *first axis direction second padding mindist* [Funzione]  
Construct a stencil by stacking *second* next to *first*. *axis* can be 0 (x-axis) or 1 (y-axis). *direction* can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with *padding* as extra space. *first* and *second* may also be '()' or #f. As opposed to **ly:stencil-combine-at-edge**, metrics are suited for successively accumulating lines of stencils. Also, *second* stencil is drawn last.  
  
If *mindist* is specified, reference points are placed apart at least by this distance. If either of the stencils is spacing, *padding* and *mindist* do not apply.
- ly:stencil-translate** *stil offset* [Funzione]  
Return a *stil*, but translated by *offset* (a pair of numbers).
- ly:stencil-translate-axis** *stil amount axis* [Funzione]  
Return a copy of *stil* but translated by *amount* in *axis* direction.

- `ly:stream-event? obj` [Funzione]  
Is *obj* a `Stream_event` object?
- `ly:string-percent-encode str` [Funzione]  
Encode all characters in string *str* with hexadecimal percent escape sequences, with the following exceptions: characters `-`, `.`, `/`, and `_`; and characters in ranges `0-9`, `A-Z`, and `a-z`.
- `ly:string-substitute a b s` [Funzione]  
Replace string *a* by string *b* in string *s*.
- `ly:system-font-load name` [Funzione]  
Load the OpenType system font *name.otf*. Fonts loaded with this command must contain three additional SFNT font tables called LILC, LILF, and LILY, needed for typesetting musical elements. Currently, only the Emmentaler and the Emmentaler-Brace fonts fulfill these requirements.  
Note that only `ly:font-get-glyph` and derived code (like `\lookup`) can access glyphs from the system fonts; text strings are handled exclusively via the Pango interface.
- `ly:text-interface::interpret-markup` [Funzione]  
Convert a text markup into a stencil. Takes three arguments, *layout*, *props*, and *markup*.  
*layout* is a `\layout` block; it may be obtained from a grob with `ly:grob-layout`. *props* is an alist chain, i.e. a list of alists. This is typically obtained with `(ly:grob-alist-chain grob (ly:output-def-lookup layout 'text-font-defaults))`. *markup* is the markup text to be processed.
- `ly:translate-cpp-warning-scheme str` [Funzione]  
Translates a string in C++ printf format and modifies it to use it for scheme formatting.
- `ly:translator? x` [Funzione]  
Is *x* a `Translator` object?
- `ly:translator-context trans` [Funzione]  
Return the context of the translator object *trans*.
- `ly:translator-description me` [Funzione]  
Return an alist of properties of translator *me*.
- `ly:translator-group? x` [Funzione]  
Is *x* a `Translator_group` object?
- `ly:translator-name trans` [Funzione]  
Return the type name of the translator object *trans*. The name is a symbol.
- `ly:transpose-key-alist l pit` [Funzione]  
Make a new key alist of *l* transposed by pitch *pit*.
- `ly:truncate-list! lst i` [Funzione]  
Take at most the first *i* of list *lst*.
- `ly:ttf->pfa ttf-file-name idx` [Funzione]  
Convert the contents of a TrueType font file to PostScript Type 42 font, returning it as a string. The optional *idx* argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of *idx* is 0.
- `ly:ttf-ps-name ttf-file-name idx` [Funzione]  
Extract the PostScript name from a TrueType font. The optional *idx* argument is useful for TrueType collections (TTC) only; it specifies the font index within the TTC. The default value of *idx* is 0.

<code>ly:undead? x</code>	[Funzione]
Is <code>x</code> a <code>Undead</code> object?	
<code>ly:unit</code>	[Funzione]
Return the unit used for lengths as a string.	
<code>ly:unpure-call data grob rest</code>	[Funzione]
Convert property <i>data</i> (unpure-pure container or procedure) to value in an unpure context defined by <i>grob</i> and possibly <i>rest</i> arguments.	
<code>ly:unpure-pure-container? x</code>	[Funzione]
Is <code>x</code> a <code>Unpure_pure_container</code> object?	
<code>ly:unpure-pure-container-pure-part pc</code>	[Funzione]
Return the pure part of <i>pc</i> .	
<code>ly:unpure-pure-container-unpure-part pc</code>	[Funzione]
Return the unpure part of <i>pc</i> .	
<code>ly:usage</code>	[Funzione]
Print usage message.	
<code>ly:verbose-output?</code>	[Funzione]
Was verbose output requested, i.e. loglevel at least <code>DEBUG</code> ?	
<code>ly:version</code>	[Funzione]
Return the current lilypond version as a list, e.g., <code>(1 3 127 uu1)</code> .	
<code>ly:warning str rest</code>	[Funzione]
A Scheme callable function to issue the warning <i>str</i> . The message is formatted with <b>format</b> and <i>rest</i> .	
<code>ly:warning-located location str rest</code>	[Funzione]
A Scheme callable function to issue the warning <i>str</i> at the specified location in an input file. The message is formatted with <b>format</b> and <i>rest</i> .	
<code>ly:wide-char-&gt;utf-8 wc</code>	[Funzione]
Encode the Unicode codepoint <i>wc</i> , an integer, as UTF-8.	

## Appendice B Schema riassuntivo

Sintassi	Descrizione	Esempio
<code>1 2 8 16</code>	durate	
<code>c4. c4..</code>	punti di aumentazione	
<code>c d e f g a b</code>	scala	
<code>fis bes</code>	alterazione	
<code>\clef treble \clef bass</code>	chiavi	
<code>\time 3/4 \time 4/4</code>	indicazione di tempo	
<code>r4 r8</code>	pausa	
<code>d ~ d</code>	legatura di valore	
<code>\key es \major</code>	armatura di chiave	

`note'`

alzare l'ottava



`note,`

abbassare l'ottava



`c( d e)`

legatura di portamento



`c\ ( c( d) e\)`

legatura di frase



`a8[ b]`

travatura



`<< \new Staff ... >>`

più righe



`c-> c-.`

articolazioni



`c2\mf c\s fz`

dinamiche



`a\< a a\!`

crescendo



`a\> a a\!`

decrescendo



`< >`

accordo



`\partial 8`

anacrusi



`\tuplet 3/2 {f g a}`

terzine



`\grace`

abbellimenti



`\lyricmode { twinkle }`

inserimento del testo twinkle  
vocale

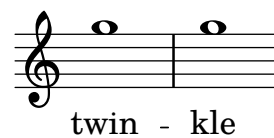
`\new Lyrics`

stampa del testo vocale

twinkle

`twin -- kle`

trattino nel testo vocale



`\chordmode { c:dim f:maj7 }`

accordi



`\new ChordNames`

mostrare i nomi degli  
accordi

C<sup>o</sup> F<sup>△</sup>

`<<{e f} \ \ {c d}>>`

polifonia



s4 s8 s16

pause spaziatrici

## Appendix C GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

### 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.



A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both

covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

#### 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements.”

## 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

#### 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

#### 11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

## ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

## Appendice D Indice dei comandi di LilyPond

Questo indice elenca tutti i comandi e le parole chiave di LilyPond con dei collegamenti alle sezioni del manuale che descrivono il loro uso. Ogni collegamento è composto da due parti. La prima parte porta al punto esatto del manuale in cui compaiono il comando o la parola chiave; la seconda parte porta all'inizio della sezione del manuale in cui compaiono il comando o la parola.

!	]
! ..... 6	] ..... 95
"	^
" " ..... 111	~ ..... 415
,	—
' ..... 2	— ..... 269
,	\
, ..... 2	\! ..... 125
—	\( ..... 136
- ..... 121	\) ..... 136
.	\< ..... 125
..... 46	\= ..... 791
/	\> ..... 125
/ ..... 415	\abs-fontsize ..... 245, 683
/+ ..... 415	\accent ..... 121
:	\accepts ..... 594, 595, 596
: ..... 165	\acciaccatura ..... 114
<	\accidentalStyle ..... 28
< ..... 167	\addChordShape ..... 373
<...> ..... 167	\addlyrics ..... 263, 265, 266
=	\addQuote ..... 211
= ..... 10	\aeolian ..... 22
>	\afterGrace ..... 115
> ..... 167	\aikenHeads ..... 41
?	\aikenHeadsMinor ..... 41
? ..... 6	\alias ..... 594
[	\allowPageTurn ..... 549
[ ..... 95	\alterBroken ..... 636
	\alternative ..... 150
	\appendToTag ..... 508
	\appoggiatura ..... 114
	\arpeggio ..... 144
	\arpeggioArrowDown ..... 144
	\arpeggioArrowUp ..... 144
	\arpeggioBracket ..... 145
	\arpeggioNormal ..... 144
	\arpeggioParenthesis ..... 145
	\arpeggioParenthesisDashed ..... 145
	\arrow-head ..... 253, 708
	\ascendens ..... 446, 453
	\auctum ..... 446, 453
	\augmentum ..... 453
	\auto-footnote ..... 730
	\autoBeamOff ..... 83, 330
	\autoBeamOn ..... 83
	\autoBreaksOff ..... 543
	\autoBreaksOn ..... 543
	\autochange ..... 328

<code>\autoLineBreaksOff</code> .....	543	<code>\descendens</code> .....	446, 453
<code>\autoLineBreaksOn</code> .....	543	<code>\dim</code> .....	126
<code>\autoPageBreaksOff</code> .....	546	<code>\dimHairpin</code> .....	126
<code>\autoPageBreaksOn</code> .....	546	<code>\dimTextDecr</code> .....	126
<code>\backslashed-digit</code> .....	730	<code>\dimTextDecresc</code> .....	126
<code>\balloonGrobText</code> .....	231	<code>\dimTextDim</code> .....	126
<code>\balloonLengthOff</code> .....	231	<code>\dir-column</code> .....	694
<code>\balloonLengthOn</code> .....	231	<code>\discant</code> .....	725
<code>\balloonText</code> .....	231	<code>\displayLilyMusic</code> .....	526
<code>\bar</code> .....	99, 105	<code>\divisioMaior</code> .....	445
<code>\barNumberCheck</code> .....	111	<code>\divisioMaxima</code> .....	445
<code>\beam</code> .....	708	<code>\divisioMinima</code> .....	445
<code>\beamExceptions</code> .....	86	<code>\dorian</code> .....	22
<code>\bendAfter</code> .....	139	<code>\dotsDown</code> .....	46
<code>\bold</code> .....	245, 683	<code>\dotsNeutral</code> .....	46
<code>\book</code> .....	472, 475	<code>\dotsUp</code> .....	46
<code>\bookOutputName</code> .....	474	<code>\doubleflat</code> .....	717
<code>\bookOutputSuffix</code> .....	474	<code>\doublesharp</code> .....	717
<code>\bookpart</code> .....	473, 475, 547	<code>\downbow</code> .....	121, 336
<code>\box</code> .....	251, 684	<code>\downmordent</code> .....	121
<code>\bracket</code> .....	130, 251, 708	<code>\downprall</code> .....	121
<code>\break</code> .....	543	<code>\draw-circle</code> .....	253, 709
<code>\breathe</code> .....	137	<code>\draw-dashed-line</code> .....	709
<code>\breve</code> .....	45, 57	<code>\draw-dotted-line</code> .....	710
<code>\cadenzaOff</code> .....	75	<code>\draw-hline</code> .....	710
<code>\cadenzaOn</code> .....	75	<code>\draw-line</code> .....	253, 710
<code>\caesura</code> .....	445	<code>\draw-squiggle-line</code> .....	711
<code>\caps</code> .....	684	<code>\drummode</code> .....	190
<code>\cavum</code> .....	446, 453	<code>\dynamic</code> .....	130, 684
<code>\center-align</code> .....	248, 693	<code>\dynamicDown</code> .....	127
<code>\center-column</code> .....	249, 693	<code>\dynamicNeutral</code> .....	127
<code>\change</code> .....	326	<code>\dynamicUp</code> .....	127
<code>\char</code> .....	731	<code>\easyHeadsOff</code> .....	39
<code>\chordmode</code> .....	5, 14, 370	<code>\easyHeadsOn</code> .....	39
<code>\chordRepeats</code> .....	343	<code>\ellipse</code> .....	711
<code>\chords</code> .....	417	<code>\epsfile</code> .....	253, 712
<code>\circle</code> .....	251, 709	<code>\espressivo</code> .....	121, 126
<code>\clef</code> .....	17	<code>\etc</code> .....	641
<code>\cm</code> .....	612	<code>\eyeglasses</code> .....	731
<code>\coda</code> .....	121	<code>\f</code> .....	124
<code>\column</code> .....	249, 693	<code>\featherDurations</code> .....	98
<code>\column-lines</code> .....	737	<code>\fermata</code> .....	121, 717
<code>\combine</code> .....	253, 694	<code>\fermataMarkup</code> .....	62, 63, 121
<code>\compound-meter</code> .....	716	<code>\ff</code> .....	124
<code>\compoundMeter</code> .....	78	<code>\fff</code> .....	124
<code>\compressMMRests</code> .....	61, 63	<code>\ffff</code> .....	124
<code>\concat</code> .....	694	<code>\fffff</code> .....	124
<code>\consists</code> .....	594	<code>\fill-line</code> .....	250, 695
<code>\context</code> .....	581, 589	<code>\fill-with-pattern</code> .....	695
<code>\cr</code> .....	125	<code>\filled-box</code> .....	253, 712
<code>\cresc</code> .....	126	<code>\finalis</code> .....	445
<code>\crescHairpin</code> .....	126	<code>\finger</code> .....	224, 684
<code>\crescTextCresc</code> .....	126	<code>\first-visible</code> .....	731
<code>\crossStaff</code> .....	330	<code>\fixed</code> .....	2
<code>\cueClef</code> .....	214	<code>\flageolet</code> .....	121
<code>\cueDuring</code> .....	214	<code>\flat</code> .....	717
<code>\cueDuringWithClef</code> .....	214	<code>\flexa</code> .....	453
<code>\customTabClef</code> .....	717	<code>\fontCaps</code> .....	684
<code>\decr</code> .....	125	<code>\fontsize</code> .....	245, 685
<code>\decresc</code> .....	126	<code>\footnote</code> .....	490, 731
<code>\defaultchild</code> .....	597	<code>\fp</code> .....	124
<code>\defaultTimeSignature</code> .....	66	<code>\fraction</code> .....	731
<code>\defineBarLine</code> .....	103	<code>\freeBass</code> .....	726
<code>\deminutum</code> .....	446, 453	<code>\frenchChords</code> .....	421
<code>\denies</code> .....	594, 595, 596	<code>\fret-diagram</code> .....	360, 722



<code>\fret-diagram-terse</code> .....	362, 722	<code>\lydian</code> .....	22
<code>\fret-diagram-verbose</code> .....	364, 723	<code>\lyricmode</code> .....	262, 263
<code>\fromproperty</code> .....	732	<code>\lyricsto</code> .....	263, 265, 266
<code>\funkHeads</code> .....	41	<code>\magnify</code> .....	245, 686
<code>\funkHeadsMinor</code> .....	41	<code>\magnifyMusic</code> .....	220
<code>\general-align</code> .....	249, 696	<code>\major</code> .....	22
<code>\germanChords</code> .....	421	<code>\makeClusters</code> .....	172
<code>\glissando</code> .....	140	<code>\map-markup-commands</code> .....	737
<code>\grace</code> .....	114	<code>\marcato</code> .....	121
<code>\halfopen</code> .....	121	<code>\mark</code> .....	112, 239
<code>\halign</code> .....	248, 697	<code>\markalphabet</code> .....	733
<code>\harmonic</code> .....	337, 346	<code>\markLengthOff</code> .....	71, 240
<code>\harmonicByFret</code> .....	346	<code>\markLengthOn</code> .....	71, 240
<code>\harmonicByRatio</code> .....	346	<code>\markletter</code> .....	733
<code>\harmonicsOff</code> .....	337	<code>\markup</code> .....	239, 241, 242, 243
<code>\harmonicsOn</code> .....	337	<code>\markuplist</code> .....	242, 256, 257
<code>\harp-pedal</code> .....	724	<code>\maxima</code> .....	45, 57
<code>\hbracket</code> .....	251, 712	<code>\medium</code> .....	686
<code>\hcenter-in</code> .....	698	<code>\melisma</code> .....	270
<code>\header</code> .....	475	<code>\melismaEnd</code> .....	270
<code>\hide</code> .....	619	<code>\mergeDifferentlyDottedOff</code> .....	177
<code>\hideKeySignature</code> .....	400	<code>\mergeDifferentlyDottedOn</code> .....	177
<code>\hideNotes</code> .....	226	<code>\mergeDifferentlyHeadedOff</code> .....	177
<code>\hideSplitTiedTabNotes</code> .....	345	<code>\mergeDifferentlyHeadedOn</code> .....	177
<code>\hideStaffSwitch</code> .....	330	<code>\mf</code> .....	124
<code>\hspace</code> .....	698	<code>\midi</code> .....	475, 579
<code>\huge</code> .....	220, 247, 685	<code>\minor</code> .....	22
<code>\improvisationOff</code> .....	44, 81	<code>\mixolydian</code> .....	22
<code>\improvisationOn</code> .....	44, 81	<code>\mm</code> .....	612
<code>\in</code> .....	612	<code>\modalInversion</code> .....	16
<code>\incipit</code> .....	457	<code>\modalTranspose</code> .....	15
<code>\inclinatum</code> .....	446, 453	<code>\mordent</code> .....	121
<code>\include</code> .....	502	<code>\mp</code> .....	124
<code>\inherit-acceptability</code> .....	595	<code>\musicglyph</code> .....	113, 717
<code>\inStaffSegno</code> .....	153	<code>\name</code> .....	594
<code>\inversion</code> .....	14	<code>\natural</code> .....	718
<code>\ionian</code> .....	22	<code>\new</code> .....	581
<code>\italianChords</code> .....	421	<code>\newSpacingSection</code> .....	566
<code>\italic</code> .....	245, 685	<code>\noBeam</code> .....	95
<code>\justified-lines</code> .....	256, 737	<code>\noBreak</code> .....	543
<code>\justify</code> .....	250, 700	<code>\noPageBreak</code> .....	546
<code>\justify-field</code> .....	699	<code>\noPageTurn</code> .....	549
<code>\justify-line</code> .....	699	<code>\normal-size-sub</code> .....	686
<code>\justify-string</code> .....	700	<code>\normal-size-super</code> .....	246, 687
<code>\keepWithTag</code> .....	505	<code>\normal-text</code> .....	687
<code>\key</code> .....	22, 41	<code>\normalsize</code> .....	220, 247, 687
<code>\killCues</code> .....	218	<code>\note</code> .....	718
<code>\label</code> .....	499	<code>\note-by-number</code> .....	718
<code>\laissezVibrer</code> .....	55	<code>\null</code> .....	248, 733
<code>\large</code> .....	220, 247, 685	<code>\number</code> .....	688
<code>\larger</code> .....	245, 247, 686	<code>\numericTimeSignature</code> .....	66
<code>\layout</code> .....	475, 539, 579, 589	<code>\octaveCheck</code> .....	10
<code>\left-align</code> .....	248, 701	<code>\omit</code> .....	619
<code>\left-brace</code> .....	732	<code>\on-the-fly</code> .....	488, 733
<code>\left-column</code> .....	701	<code>\once</code> .....	605
<code>\lheel</code> .....	121	<code>\oneVoice</code> .....	173
<code>\line</code> .....	701	<code>\open</code> .....	121, 336
<code>\linea</code> .....	446, 453	<code>\oriscus</code> .....	446, 453
<code>\lineprall</code> .....	121	<code>\ottava</code> .....	24
<code>\locrian</code> .....	22	<code>\oval</code> .....	713
<code>\longa</code> .....	45, 57	<code>\overlay</code> .....	702
<code>\longfermata</code> .....	121	<code>\override</code> .....	603, 607, 733
<code>\lookup</code> .....	732	<code>\override-lines</code> .....	737
<code>\lower</code> .....	248, 702	<code>\overrideProperty</code> .....	607
<code>\ltoe</code> .....	121	<code>\overrideTimeSignatureSettings</code> .....	66

<code>\overtie</code>	688	<code>\rest-by-number</code>	719
<code>\p</code>	124	<code>\retrograde</code>	14
<code>\pad-around</code>	252, 702	<code>\reverseturn</code>	121
<code>\pad-markup</code>	252, 702	<code>\revert</code>	604
<code>\pad-to-box</code>	252, 703	<code>\revertTimeSignatureSettings</code>	68
<code>\pad-x</code>	252, 703	<code>\rfz</code>	124
<code>\page-link</code>	734	<code>\rheel</code>	121
<code>\page-ref</code>	499, 734	<code>\right-align</code>	248, 704
<code>\pageBreak</code>	546	<code>\right-brace</code>	734
<code>\pageTurn</code>	549	<code>\right-column</code>	704
<code>\paper</code>	475, 529	<code>\rightHandFinger</code>	382
<code>\parallelMusic</code>	187	<code>\roman</code>	688
<code>\parenthesize</code>	229, 713	<code>\romanStringNumbers</code>	336
<code>\partcombine</code>	181, 290	<code>\rotate</code>	704
<code>\partcombineApart</code>	183	<code>\rounded-box</code>	251, 715
<code>\partcombineAutomatic</code>	183	<code>\rtoe</code>	121
<code>\partcombineChords</code>	183	<code>\sacredHarpHeads</code>	41
<code>\partcombineSoloI</code>	183	<code>\sacredHarpHeadsMinor</code>	41
<code>\partcombineSoloII</code>	183	<code>\sans</code>	689
<code>\partcombineUnisono</code>	183	<code>\scale</code>	715
<code>\partial</code>	73, 150, 152	<code>\scaleDurations</code>	53, 76
<code>\path</code>	714	<code>\score</code>	471, 475, 720
<code>\pattern</code>	734	<code>\score-lines</code>	738
<code>\pes</code>	453	<code>\segno</code>	121
<code>\phrasingSlurDashed</code>	136	<code>\semiflat</code>	721
<code>\phrasingSlurDashPattern</code>	136	<code>\semiGermanChords</code>	421
<code>\phrasingSlurDotted</code>	136	<code>\semisharp</code>	721
<code>\phrasingSlurDown</code>	136	<code>\sesquiflat</code>	721
<code>\phrasingSlurHalfDashed</code>	136	<code>\sesquisharp</code>	721
<code>\phrasingSlurHalfSolid</code>	136	<code>\set</code>	86, 601, 607
<code>\phrasingSlurNeutral</code>	136	<code>\sf</code>	124
<code>\phrasingSlurSolid</code>	136	<code>\sff</code>	124
<code>\phrasingSlurUp</code>	136	<code>\sfz</code>	124
<code>\phrygian</code>	22	<code>\shape</code>	633
<code>\pitchedTrill</code>	149	<code>\sharp</code>	721
<code>\portato</code>	121	<code>\shiftOff</code>	177
<code>\postscript</code>	253, 714	<code>\shiftOn</code>	177
<code>\powerChords</code>	385	<code>\shiftOnn</code>	177
<code>\pp</code>	124	<code>\shiftOnnn</code>	177
<code>\ppp</code>	124	<code>\shortfermata</code>	121
<code>\pppp</code>	124	<code>\showKeySignature</code>	400
<code>\ppppp</code>	124	<code>\showStaffSwitch</code>	330
<code>\prall</code>	121	<code>\signumcongruentiae</code>	121
<code>\pralldown</code>	121	<code>\simple</code>	689
<code>\prallmordent</code>	121	<code>\skip</code>	60, 286
<code>\prallprall</code>	121	<code>\slashed-digit</code>	735
<code>\prallup</code>	121	<code>\slashedGrace</code>	114
<code>\predefinedFretboardsOff</code>	380	<code>\slurDashed</code>	133
<code>\predefinedFretboardsOn</code>	380	<code>\slurDashPattern</code>	134
<code>\property-recursive</code>	734	<code>\slurDotted</code>	133
<code>\pt</code>	612	<code>\slurDown</code>	133
<code>\pushToTag</code>	508	<code>\slurHalfDashed</code>	133
<code>\put-adjacent</code>	703	<code>\slurHalfSolid</code>	133
<code>\quilisma</code>	446, 453	<code>\slurNeutral</code>	133
<code>\quoteDuring</code>	211, 214	<code>\slurSolid</code>	133
<code>\raise</code>	248, 703	<code>\slurUp</code>	134
<code>\relative</code>	2, 5, 14, 329	<code>\small</code>	220, 247, 689
<code>\RemoveEmptyStaves</code>	204, 205	<code>\smallCaps</code>	689
<code>\removeWithTag</code>	505	<code>\smaller</code>	245, 247, 690
<code>\repeat</code>	150	<code>\snappizzicato</code>	121
<code>\repeat percent</code>	162	<code>\sostenutoOff</code>	332
<code>\repeat tremolo</code>	165	<code>\sostenutoOn</code>	332
<code>\repeatTie</code>	55, 153, 287	<code>\southernHarmonyHeads</code>	41
<code>\replace</code>	688	<code>\southernHarmonyHeadsMinor</code>	41
<code>\rest</code>	57, 719	<code>\sp</code>	124



## A

absolute.....	781
accepts.....	594
acciaccatura.....	781
accidentalStyle.....	781
addChordShape.....	373, 781
addInstrumentDefinition.....	781
additionalPitchPrefix.....	419
addQuote.....	211, 781
aeolian.....	22
afterGrace.....	115, 781
aikenHeads.....	41
aikenHeadsMinor.....	41
alias.....	594
alignAboveContext.....	597
alignBelowContext.....	284, 597
allowPageTurn.....	781
allowVoltaHook.....	781
alterBroken.....	781
annotate-spacing.....	576
appendToTag.....	781
applyContext.....	781
applyMusic.....	781
applyOutput.....	781
appoggiatura.....	781
arpeggio.....	144
arpeggioArrowDown.....	144
arpeggioArrowUp.....	144
arpeggioBracket.....	145
arpeggioNormal.....	144
arpeggioParenthesis.....	145
arpeggioParenthesisDashed.....	145
arrow-head.....	253
assertBeamQuant.....	782
assertBeamSlope.....	782
aug.....	412
auto-first-page-number.....	538
autoBeaming.....	86, 579
autoBeamOff.....	83
autoBeamOn.....	83
autochange.....	328, 782

## B

Balloon_engraver.....	231
balloonGrobText.....	231, 782
balloonLengthOff.....	231
balloonLengthOn.....	231
balloonText.....	231, 782
banjo-c-tuning.....	388
banjo-modal-tuning.....	388
banjo-open-d-tuning.....	388
banjo-open-dm-tuning.....	388
bar.....	99, 105, 782
barCheckSynchronize.....	111
BarNumber.....	106
barNumberCheck.....	111, 782
barNumberVisibility.....	106
bartype.....	105
base-shortest-duration.....	565
baseMoment.....	86
beamExceptions.....	86, 782
beatStructure.....	86
bendAfter.....	139, 782

binding-offset.....	535
blank-after-score-page-penalty.....	537
blank-last-page-penalty.....	537
blank-page-penalty.....	537
bold.....	245
bookOutputName.....	782
bookOutputSuffix.....	782
bookTitleMarkup.....	485
bottom-margin.....	531
box.....	251
bracket.....	130, 251, 332
breakable.....	84
breathe.....	137, 782
breve.....	45, 57

## C

cadenzaOff.....	75
cadenzaOn.....	75
center-align.....	248
center-column.....	249
change.....	326
check-consistency.....	534
chordChanges.....	417
chordmode.....	5, 14, 370
chordNameExceptions.....	420
chordNameLowercaseMinor.....	419
ChordNames.....	370
chordNameSeparator.....	420
chordNoteNamer.....	420
chordPrefixSpacer.....	421
chordRepeats.....	782
chordRootNamer.....	419
circle.....	251
clef.....	17, 782
clip-regions.....	511
color.....	227
column.....	249
combine.....	253
common-shortest-duration.....	565
Completion_heads_engraver.....	79
Completion_rest_engraver.....	79
compoundMeter.....	782
compressMMRests.....	61, 63, 782
consists.....	594
controlpitch.....	10
cr.....	125
cresc.....	126
crescHairpin.....	126
crescTextCresc.....	126
cross.....	38
crossStaff.....	783
cueClef.....	214, 783
cueClefUnset.....	783
cueDuring.....	214, 783
cueDuringWithClef.....	214, 783
currentBarNumber.....	106, 120

## D

deadNote.....	783
decr.....	125
decreasc.....	126
default.....	28, 29
default-staff-staff-spacing.....	550
defaultBarType.....	105
defaultNoteHeads.....	783
defaultTimeSignature.....	66
defineBarLine.....	103, 783
denies.....	594
dim.....	126, 412
dimHairpin.....	126
dimTextDecr.....	126
dimTextDecresc.....	126
dimTextDim.....	126
displayLilyMusic.....	783
displayMusic.....	783
displayScheme.....	783
dodecaphonic.....	32
dodecaphonic-first.....	33
dodecaphonic-no-repeat.....	33
dorian.....	22
dotsDown.....	46
dotsNeutral.....	46
dotsUp.....	46
draw-circle.....	253
draw-line.....	253
drummode.....	190
DrumStaff.....	190
dynamic.....	130
dynamicDown.....	127
DynamicLineSpanner.....	127
dynamicNeutral.....	127
dynamicUp.....	127

## E

easyHeadsOff.....	39
easyHeadsOn.....	39
endSpanners.....	783
epsfile.....	253
espressivo.....	126
eventChords.....	783
extra-offset.....	550

## F

f.....	124
featherDurations.....	98, 783
fermataMarkup.....	62, 63
ff.....	124
fff.....	124
ffff.....	124
fffff.....	124
fill-line.....	250
filled-box.....	253
finger.....	224, 783
first-page-number.....	538
fixed.....	2, 783
followVoice.....	330
font-interface.....	223, 257
font-size.....	220, 223
fontsize.....	245

fontSize.....	220
footnote.....	783
forget.....	34
four-string-banjo.....	388
fp.....	124
fret-diagram.....	360
fret-diagram-interface.....	366
fret-diagram-terse.....	362
fret-diagram-verbose.....	364
FretBoards.....	369
funkHeads.....	41
funkHeadsMinor.....	41

## G

general-align.....	249
glissando.....	140
grace.....	784
GregorianTranscriptionStaff.....	190
Grid_line_span_engraver.....	232
Grid_point_engraver.....	232
gridInterval.....	232
grobdescriptions.....	784
grow-direction.....	98

## H

halign.....	248
harmonicByFret.....	784
harmonicByRatio.....	784
harmonicNote.....	784
harmonicsOn.....	784
hbracket.....	251
hide.....	784
hideKeySignature.....	400
hideNotes.....	226
hideStaffSwitch.....	330
horizontal-shift.....	536
Horizontal_bracket_engraver.....	234
huge.....	220, 247

## I

improvisationOff.....	44, 81
improvisationOn.....	44, 81
incipit.....	457, 784
indent.....	208, 536, 569
inherit-acceptability.....	784
inner-margin.....	535
inStaffSegno.....	784
instrumentSwitch.....	784
inversion.....	784
ionian.....	22
italic.....	245

## J

justified-lines.....	256
justify.....	250

**K**

keepWithTag	785
key	22, 41, 785
killCues	218, 785

**L**

label	785
laissezVibrer	55
language	785
languageRestore	785
languageSaveAndChange	785
large	220, 247
larger	245, 247
last-bottom-spacing	533
layout file	541
layout-set-staff-size	541
left-align	248
left-margin	534
line-width	534, 569
locrian	22
longa	45, 57
lower	248
ly:minimal-breaking	548
ly:one-line-breaking	548
ly:optimal-breaking	548
ly:page-turn-breaking	548
lydian	22

**M**

m	412
magnification->font-size	220, 541
magnify	245
magnifyMusic	220, 785
magnifyStaff	785
magstep	220, 541, 612
maj	412
major	22
major seven symbols	421
majorSevenSymbol	419
make-dynamic-script	131
make-pango-font-tree	259
makeClusters	172, 785
makeDefaultStringTuning	785
mark	112, 239, 785
markLengthOff	71, 240
markLengthOn	71, 240
markup	239, 241, 242, 243
markup-markup-spacing	533
markup-system-spacing	533
markuplist	242, 256, 257
markupMap	785
max-systems-per-page	536
maxima	45, 57
measureLength	86, 120
measurePosition	73, 120
MensuralStaff	190
mergeDifferentlyDottedOff	177
mergeDifferentlyDottedOn	177
mergeDifferentlyHeadedOff	177
mergeDifferentlyHeadedOn	177
mf	124
midiBalance	524

midiChannelMapping	521
midiChorusLevel	524
midiExpression	524
midiPanPosition	524
midiReverbLevel	524
min-systems-per-page	536
minimum-Y-extent	550
minimumFret	343, 381
minimumPageTurnLength	548
minimumRepeatLengthForPageTurn	549
minor	22
minorChordModifier	420
mixed	332
mixolydian	22
modalInversion	16, 785
modalTranspose	15, 785
modern	30
modern-cautionary	30
modern-voice	31
modern-voice-cautionary	31
mp	124
MultiMeasureRestText	62
musicglyph	113
musicMap	786

**N**

name	594
neo-modern	32
neo-modern-cautionary	32
neo-modern-voice	32
neo-modern-voice-cautionary	32
no-reset	33
noBeam	95
nonstaff-nonstaff-spacing	550
nonstaff-relatedstaff-spacing	550
nonstaff-unrelatedstaff-spacing	550
noPageBreak	786
noPageTurn	786
normal-size-super	246
normalsize	220, 247
Note_heads_engraver	79
null	248
numericTimeSignature	66

**O**

octaveCheck	10, 786
offset	786
omit	786
once	786
oneVoice	173
ottava	24, 786
outer-margin	535
outside-staff-horizontal-padding	563
outside-staff-padding	563
outside-staff-priority	563
overrideProperty	786
overrideTimeSignatureSettings	786

## P

p.....	124
pad-around.....	252
pad-markup.....	252
pad-to-box.....	252
pad-x.....	252
page-breaking.....	537
page-breaking-system-system-spacing.....	537
page-count.....	537
page-number-type.....	538
page-spacing-weight.....	538
pageBreak.....	786
pageTurn.....	786
palmMute.....	786
palmMuteOn.....	787
paper-height.....	531
paper-width.....	534
parallelMusic.....	187, 787
parenthesize.....	229, 787
partcombine.....	181, 787
partcombineApart.....	183
partcombineAutomatic.....	183
partcombineChords.....	183
partcombineDown.....	787
partcombineForce.....	787
partcombineSoloI.....	183
partcombineSoloII.....	183
partcombineUnisono.....	183
partcombineUp.....	787
partial.....	73, 787
pedalSustainStyle.....	332
percent.....	162
phrasingSlurDashed.....	136
phrasingSlurDashPattern.....	136, 787
phrasingSlurDotted.....	136
phrasingSlurDown.....	136
phrasingSlurHalfDashed.....	136
phrasingSlurHalfSolid.....	136
phrasingSlurNeutral.....	136
phrasingSlurSolid.....	136
phrasingSlurUp.....	136
phrygian.....	22
piano.....	31
piano-cautionary.....	31
PianoStaff.....	325, 328
Pitch_squash_engraver.....	81
pitchedTrill.....	149, 787
pointAndClickOff.....	787
pointAndClickOn.....	788
pointAndClickTypes.....	788
postscript.....	253
powerChords.....	385
pp.....	124
ppp.....	124
pppp.....	124
ppppp.....	124
predefinedFretboardsOff.....	380
predefinedFretboardsOn.....	380
print-all-headers.....	538
print-first-page-number.....	538
print-page-number.....	538
propertyOverride.....	788
propertyRevert.....	788
propertySet.....	788

propertyTweak.....	788
propertyUnset.....	788
pushToTag.....	788

## Q

quotedCueEventTypes.....	213
quotedEventTypes.....	213
quoteDuring.....	211, 214, 788

## R

r.....	57
R.....	61
ragged-bottom.....	531
ragged-last.....	535, 569
ragged-last-bottom.....	531
ragged-right.....	535, 569
raise.....	248
relative.....	2, 5, 14, 329, 788
removeWithTag.....	788
repeatCommands.....	158
repeatTie.....	55
resetRelativeOctave.....	789
rest.....	57
restrainOpenStrings.....	343
retrograde.....	14, 789
revertTimeSignatureSettings.....	789
rfz.....	124
rgb-color.....	228
RhythmicStaff.....	190
right-align.....	248
right-margin.....	534
rightHandFinger.....	382, 789
rounded-box.....	251

## S

s.....	60
sacredHarpHeads.....	41
sacredHarpHeadsMinor.....	41
scaleDurations.....	53, 76, 789
score-markup-spacing.....	533
score-system-spacing.....	533
scoreTitleMarkup.....	485
self-alignment-X.....	550
set.....	86
set-global-staff-size.....	541
set-octavation.....	24
settingsFrom.....	789
sf.....	124
sff.....	124
sfz.....	124
shape.....	789
shiftDurations.....	789
shiftOff.....	177
shiftOn.....	177
shiftOnn.....	177
shiftOnnn.....	177
short-indent.....	208, 536
show-available-fonts.....	259
showFirstLength.....	512
showKeySignature.....	400
showLastLength.....	512

showStaffSwitch.....	330
single.....	789
skip.....	60, 789
skipTypesetting.....	512
slashChordSeparator.....	420
slashedGrace.....	789
slurDashed.....	133
slurDashPattern.....	134, 789
slurDotted.....	133
slurDown.....	133
slurHalfDashed.....	133
slurHalfSolid.....	133
slurNeutral.....	133
slurSolid.....	133
slurUp.....	134
small.....	220, 247
smaller.....	245, 247
sostenutoOff.....	332
sostenutoOn.....	332
southernHarmonyHeads.....	41
southernHarmonyHeadsMinor.....	41
sp.....	124
spacing.....	565
spacingTweaks.....	789
Span_stem_engraver.....	330
spp.....	124
staff-affinity.....	550
staff-staff-spacing.....	550
Staff_midiInstrument.....	525
Staff_symbol_engraver.....	204
staffgroup-staff-spacing.....	550
start-repeat.....	158
startGroup.....	234
startStaff.....	197, 200
startTrillSpan.....	148
Stem.....	330
stem-spacing-correction.....	565
stemDown.....	229
stemLeftBeamCount.....	96
stemNeutral.....	229
stemRightBeamCount.....	96
stemUp.....	229
stopGroup.....	234
stopStaff.....	197, 200, 204
stopTrillSpan.....	148
storePredefinedDiagram.....	373, 789
stringTuning.....	356, 789
stringTunings.....	356, 369
styledNoteHeads.....	789
sub.....	246
suggestAccidentals.....	440
super.....	246
sus.....	415
sustainOff.....	332
sustainOn.....	332
system-count.....	536
system-separator-markup.....	538
system-system-spacing.....	533
systems-per-page.....	536

## T

tabChordRepeats.....	790
tabChordRepetition.....	790
TabStaff.....	190, 341
TabVoice.....	341
tag.....	790
tagGroup.....	790
taor.....	400
teaching.....	33
teeny.....	220, 247
tempo.....	70
temporary.....	790
text.....	332
textLengthOff.....	63, 236
textLengthOn.....	63, 236
textSpannerDown.....	237
textSpannerNeutral.....	237
textSpannerUp.....	237
thumb.....	224
tieDashed.....	55
tieDashPattern.....	790
tieDotted.....	55
tieDown.....	55
tieNeutral.....	55
tieSolid.....	55
tieUp.....	55
time.....	65, 86, 790
times.....	790
timeSignatureFraction.....	76
tiny.....	220, 247
tocItem.....	790
top-margin.....	531
top-markup-spacing.....	533
top-system-spacing.....	533
translate.....	249
translate-scaled.....	249
transpose.....	5, 11, 14, 790
transposedCueDuring.....	218, 790
transposition.....	26, 211, 790
treCorde.....	332
tremolo.....	165
triangle.....	253
trill.....	148
tuplet.....	48, 76, 790
tupletDown.....	48
tupletNeutral.....	48
TupletNumber.....	49
tupletNumberFormatFunction.....	49
tupletSpan.....	791
tupletSpannerDuration.....	49
tupletUp.....	48
tweak.....	791
two-sided.....	535
type.....	594

## U

unaCorda.....	332
underline.....	245
undo.....	791
unfold.....	160
unfoldRepeats.....	791
unHideNotes.....	226



## V

VaticanaStaff.....	190
VerticalAxisGroup.....	550
voice.....	28, 30
Voice.....	173
voiceOne.....	173
void.....	791

## W

walkerHeads.....	41
walkerHeadsMinor.....	41
whichBar.....	105
with-color.....	227

withMusicProperty.....	791
wordwrap.....	250
wordwrap-lines.....	256

## X

X-offset.....	550
x11-color.....	227, 228
xNote.....	791
xNotesOn.....	791

## Appendice E Indice di LilyPond

Oltre a tutti i comandi e le parole chiave di LilyPond, questo indice elenca i termini musicali e le espressioni che si riferiscono a ognuno di essi, corredati di collegamenti alle relative sezioni del manuale. Ogni collegamento è composto da due parti. La prima parte porta al punto esatto del manuale in cui compare l'argomento; la seconda parte porta all'inizio della sezione del manuale in cui l'argomento è trattato.

!	]
! ..... 6	] ..... 95
"	^
" " ..... 111	^ ..... 415
,	-
' ..... 2	- ..... 269
,	\
, ..... 2	\! ..... 125
—	\( ..... 136
- ..... 121	\) ..... 136
•	\< ..... 125
• ..... 46	\= ..... 791
/	\> ..... 125
/ ..... 415	\abs-fontsize ..... 245, 683
/+ ..... 415	\accent ..... 121
:	\accepts ..... 594, 595, 596
: ..... 165	\acciaccatura ..... 114
<	\accidentalStyle ..... 28
< ..... 167	\addChordShape ..... 373
<...> ..... 167	\addlyrics ..... 263, 265, 266
=	\addQuote ..... 211
= ..... 10	\aeolian ..... 22
>	\afterGrace ..... 115
> ..... 167	\aikenHeads ..... 41
?	\aikenHeadsMinor ..... 41
? ..... 6	\alias ..... 594
[	\allowPageTurn ..... 549
[ ..... 95	\alterBroken ..... 636
	\alternative ..... 150
	\appendToTag ..... 508
	\appoggiatura ..... 114
	\arpeggio ..... 144
	\arpeggioArrowDown ..... 144
	\arpeggioArrowUp ..... 144
	\arpeggioBracket ..... 145
	\arpeggioNormal ..... 144
	\arpeggioParenthesis ..... 145
	\arpeggioParenthesisDashed ..... 145
	\arrow-head ..... 253, 708
	\ascendens ..... 446, 453
	\auctum ..... 446, 453
	\augmentum ..... 453
	\auto-footnote ..... 730
	\autoBeamOff ..... 83, 330
	\autoBeamOn ..... 83
	\autoBreaksOff ..... 543
	\autoBreaksOn ..... 543

<code>\autochange</code> .....	328	<code>\deminutum</code> .....	446, 453
<code>\autoLineBreaksOff</code> .....	543	<code>\denies</code> .....	594, 595, 596
<code>\autoLineBreaksOn</code> .....	543	<code>\descendens</code> .....	446, 453
<code>\autoPageBreaksOff</code> .....	546	<code>\dim</code> .....	126
<code>\autoPageBreaksOn</code> .....	546	<code>\dimHairpin</code> .....	126
<code>\backslashed-digit</code> .....	730	<code>\dimTextDecr</code> .....	126
<code>\balloonGrobText</code> .....	231	<code>\dimTextDecresc</code> .....	126
<code>\balloonLengthOff</code> .....	231	<code>\dimTextDim</code> .....	126
<code>\balloonLengthOn</code> .....	231	<code>\dir-column</code> .....	694
<code>\balloonText</code> .....	231	<code>\discant</code> .....	725
<code>\bar</code> .....	99, 105	<code>\displayLilyMusic</code> .....	526
<code>\barNumberCheck</code> .....	111	<code>\divisioMaior</code> .....	445
<code>\beam</code> .....	708	<code>\divisioMaxima</code> .....	445
<code>\beamExceptions</code> .....	86	<code>\divisioMinima</code> .....	445
<code>\bendAfter</code> .....	139	<code>\dorian</code> .....	22
<code>\bold</code> .....	245, 683	<code>\dotsDown</code> .....	46
<code>\book</code> .....	472, 475	<code>\dotsNeutral</code> .....	46
<code>\bookOutputName</code> .....	474	<code>\dotsUp</code> .....	46
<code>\bookOutputSuffix</code> .....	474	<code>\doubleflat</code> .....	717
<code>\bookpart</code> .....	473, 475, 547	<code>\doublesharp</code> .....	717
<code>\box</code> .....	251, 684	<code>\downbow</code> .....	121, 336
<code>\bracket</code> .....	130, 251, 708	<code>\downmordent</code> .....	121
<code>\break</code> .....	543	<code>\downprall</code> .....	121
<code>\breathe</code> .....	137	<code>\draw-circle</code> .....	253, 709
<code>\breve</code> .....	45, 57	<code>\draw-dashed-line</code> .....	709
<code>\cadenzaOff</code> .....	75	<code>\draw-dotted-line</code> .....	710
<code>\cadenzaOn</code> .....	75	<code>\draw-hline</code> .....	710
<code>\caesura</code> .....	445	<code>\draw-line</code> .....	253, 710
<code>\caps</code> .....	684	<code>\draw-squiggle-line</code> .....	711
<code>\cavum</code> .....	446, 453	<code>\drummode</code> .....	190
<code>\center-align</code> .....	248, 693	<code>\dynamic</code> .....	130, 684
<code>\center-column</code> .....	249, 693	<code>\dynamicDown</code> .....	127
<code>\change</code> .....	326	<code>\dynamicNeutral</code> .....	127
<code>\char</code> .....	731	<code>\dynamicUp</code> .....	127
<code>\chordmode</code> .....	5, 14, 370	<code>\easyHeadsOff</code> .....	39
<code>\chordRepeats</code> .....	343	<code>\easyHeadsOn</code> .....	39
<code>\chords</code> .....	417	<code>\ellipse</code> .....	711
<code>\circle</code> .....	251, 709	<code>\epsfile</code> .....	253, 712
<code>\clef</code> .....	17	<code>\espressivo</code> .....	121, 126
<code>\cm</code> .....	612	<code>\etc</code> .....	641
<code>\coda</code> .....	121	<code>\eyeglasses</code> .....	731
<code>\column</code> .....	249, 693	<code>\f</code> .....	124
<code>\column-lines</code> .....	737	<code>\featherDurations</code> .....	98
<code>\combine</code> .....	253, 694	<code>\fermata</code> .....	121, 717
<code>\compound-meter</code> .....	716	<code>\fermataMarkup</code> .....	62, 63, 121
<code>\compoundMeter</code> .....	78	<code>\ff</code> .....	124
<code>\compressMMRests</code> .....	61, 63	<code>\fff</code> .....	124
<code>\concat</code> .....	694	<code>\ffff</code> .....	124
<code>\consists</code> .....	594	<code>\fffff</code> .....	124
<code>\context</code> .....	581, 589	<code>\fill-line</code> .....	250, 695
<code>\context in \layout block</code> .....	589	<code>\fill-with-pattern</code> .....	695
<code>\cr</code> .....	125	<code>\filled-box</code> .....	253, 712
<code>\cresc</code> .....	126	<code>\finalis</code> .....	445
<code>\crescHairpin</code> .....	126	<code>\finger</code> .....	224, 684
<code>\crescTextCresc</code> .....	126	<code>\first-visible</code> .....	731
<code>\crossStaff</code> .....	330	<code>\fixed</code> .....	2
<code>\cueClef</code> .....	214	<code>\flageolet</code> .....	121
<code>\cueDuring</code> .....	214	<code>\flat</code> .....	717
<code>\cueDuringWithClef</code> .....	214	<code>\flexa</code> .....	453
<code>\customTabClef</code> .....	717	<code>\fontCaps</code> .....	684
<code>\decr</code> .....	125	<code>\fontsize</code> .....	245, 685
<code>\decresc</code> .....	126	<code>\footnote</code> .....	490, 731
<code>\defaultchild</code> .....	597	<code>\fp</code> .....	124
<code>\defaultTimeSignature</code> .....	66	<code>\fraction</code> .....	731
<code>\defineBarLine</code> .....	103	<code>\freeBass</code> .....	726

<code>\frenchChords</code> .....	421	<code>\lower</code> .....	248, 702
<code>\fret-diagram</code> .....	360, 722	<code>\ltoe</code> .....	121
<code>\fret-diagram-terse</code> .....	362, 722	<code>\lydian</code> .....	22
<code>\fret-diagram-verbose</code> .....	364, 723	<code>\lyricmode</code> .....	262, 263
<code>\fromproperty</code> .....	732	<code>\lyricsto</code> .....	263, 265, 266
<code>\funkHeads</code> .....	41	<code>\magnify</code> .....	245, 686
<code>\funkHeadsMinor</code> .....	41	<code>\magnifyMusic</code> .....	220
<code>\general-align</code> .....	249, 696	<code>\major</code> .....	22
<code>\germanChords</code> .....	421	<code>\makeClusters</code> .....	172
<code>\glissando</code> .....	140	<code>\map-markup-commands</code> .....	737
<code>\grace</code> .....	114	<code>\marcato</code> .....	121
<code>\halfopen</code> .....	121	<code>\mark</code> .....	112, 239
<code>\halign</code> .....	248, 697	<code>\markalphabet</code> .....	733
<code>\harmonic</code> .....	337, 346	<code>\markLengthOff</code> .....	71, 240
<code>\harmonicByFret</code> .....	346	<code>\markLengthOn</code> .....	71, 240
<code>\harmonicByRatio</code> .....	346	<code>\markletter</code> .....	733
<code>\harmonicsOff</code> .....	337	<code>\markup</code> .....	239, 241, 242, 243
<code>\harmonicsOn</code> .....	337	<code>\markuplist</code> .....	242, 256, 257
<code>\harp-pedal</code> .....	724	<code>\maxima</code> .....	45, 57
<code>\hbracket</code> .....	251, 712	<code>\medium</code> .....	686
<code>\hcenter-in</code> .....	698	<code>\melisma</code> .....	270
<code>\header</code> .....	475	<code>\melismaEnd</code> .....	270
<code>\hide</code> .....	619	<code>\mergeDifferentlyDottedOff</code> .....	177
<code>\hideKeySignature</code> .....	400	<code>\mergeDifferentlyDottedOn</code> .....	177
<code>\hideNotes</code> .....	226	<code>\mergeDifferentlyHeadedOff</code> .....	177
<code>\hideSplitTiedTabNotes</code> .....	345	<code>\mergeDifferentlyHeadedOn</code> .....	177
<code>\hideStaffSwitch</code> .....	330	<code>\mf</code> .....	124
<code>\hspace</code> .....	698	<code>\midi</code> .....	475, 579
<code>\huge</code> .....	220, 247, 685	<code>\minor</code> .....	22
<code>\improvisationOff</code> .....	44, 81	<code>\mixolydian</code> .....	22
<code>\improvisationOn</code> .....	44, 81	<code>\mmm</code> .....	612
<code>\in</code> .....	612	<code>\modalInversion</code> .....	16
<code>\incipit</code> .....	457	<code>\modalTranspose</code> .....	15
<code>\inclinatum</code> .....	446, 453	<code>\mordent</code> .....	121
<code>\include</code> .....	502	<code>\mp</code> .....	124
<code>\inherit-acceptability</code> .....	595	<code>\musicglyph</code> .....	113, 717
<code>\inStaffSegno</code> .....	153	<code>\name</code> .....	594
<code>\inversion</code> .....	14	<code>\natural</code> .....	718
<code>\ionian</code> .....	22	<code>\new</code> .....	581
<code>\italianChords</code> .....	421	<code>\newSpacingSection</code> .....	566
<code>\italic</code> .....	245, 685	<code>\noBeam</code> .....	95
<code>\justified-lines</code> .....	256, 737	<code>\noBreak</code> .....	543
<code>\justify</code> .....	250, 700	<code>\noPageBreak</code> .....	546
<code>\justify-field</code> .....	699	<code>\noPageTurn</code> .....	549
<code>\justify-line</code> .....	699	<code>\normal-size-sub</code> .....	686
<code>\justify-string</code> .....	700	<code>\normal-size-super</code> .....	246, 687
<code>\keepWithTag</code> .....	505	<code>\normal-text</code> .....	687
<code>\key</code> .....	22, 41	<code>\normalsize</code> .....	220, 247, 687
<code>\killCues</code> .....	218	<code>\note</code> .....	718
<code>\label</code> .....	499	<code>\note-by-number</code> .....	718
<code>\laissezVibrer</code> .....	55	<code>\null</code> .....	248, 733
<code>\large</code> .....	220, 247, 685	<code>\number</code> .....	688
<code>\larger</code> .....	245, 247, 686	<code>\numericTimeSignature</code> .....	66
<code>\layout</code> .....	475, 539, 579, 589	<code>\octaveCheck</code> .....	10
<code>\left-align</code> .....	248, 701	<code>\omit</code> .....	619
<code>\left-brace</code> .....	732	<code>\on-the-fly</code> .....	488, 733
<code>\left-column</code> .....	701	<code>\once</code> .....	603
<code>\lheel</code> .....	121	<code>\once</code> .....	605
<code>\line</code> .....	701	<code>\oneVoice</code> .....	173
<code>\linea</code> .....	446, 453	<code>\open</code> .....	121, 336
<code>\lineprall</code> .....	121	<code>\oriscus</code> .....	446, 453
<code>\locrian</code> .....	22	<code>\ottava</code> .....	24
<code>\longa</code> .....	45, 57	<code>\oval</code> .....	713
<code>\longfermata</code> .....	121	<code>\overlay</code> .....	702
<code>\lookup</code> .....	732	<code>\override</code> .....	603, 607, 733

<code>\override-lines</code> .....	737	<code>\repeat percent</code> .....	162
<code>\overrideProperty</code> .....	607	<code>\repeat tremolo</code> .....	165
<code>\overrideTimeSignatureSettings</code> .....	66	<code>\repeatTie</code> .....	55, 153, 287
<code>\overtie</code> .....	688	<code>\replace</code> .....	688
<code>\p</code> .....	124	<code>\rest</code> .....	57, 719
<code>\pad-around</code> .....	252, 702	<code>\rest-by-number</code> .....	719
<code>\pad-markup</code> .....	252, 702	<code>\retrograde</code> .....	14
<code>\pad-to-box</code> .....	252, 703	<code>\reverseturn</code> .....	121
<code>\pad-x</code> .....	252, 703	<code>\revert</code> .....	604
<code>\page-link</code> .....	734	<code>\revertTimeSignatureSettings</code> .....	68
<code>\page-ref</code> .....	499, 734	<code>\rfz</code> .....	124
<code>\pageBreak</code> .....	546	<code>\rheel</code> .....	121
<code>\pageTurn</code> .....	549	<code>\right-align</code> .....	248, 704
<code>\paper</code> .....	475, 529	<code>\right-brace</code> .....	734
<code>\parallelMusic</code> .....	187	<code>\right-column</code> .....	704
<code>\parenthesize</code> .....	229, 713	<code>\rightHandFinger</code> .....	382
<code>\partcombine</code> .....	181, 290	<code>\roman</code> .....	688
<code>\partcombine e testo cantato</code> .....	290	<code>\romanStringNumbers</code> .....	336
<code>\partcombine e testo vocale</code> .....	184	<code>\rotate</code> .....	704
<code>\partcombineApart</code> .....	183	<code>\rounded-box</code> .....	251, 715
<code>\partcombineAutomatic</code> .....	183	<code>\rtoe</code> .....	121
<code>\partcombineChords</code> .....	183	<code>\sacredHarpHeads</code> .....	41
<code>\partcombineSoloI</code> .....	183	<code>\sacredHarpHeadsMinor</code> .....	41
<code>\partcombineSoloII</code> .....	183	<code>\sans</code> .....	689
<code>\partcombineUnisono</code> .....	183	<code>\scale</code> .....	715
<code>\partial</code> .....	73, 150, 152	<code>\scaleDurations</code> .....	53, 76
<code>\path</code> .....	714	<code>\score</code> .....	471, 475, 720
<code>\pattern</code> .....	734	<code>\score-lines</code> .....	738
<code>\pes</code> .....	453	<code>\segno</code> .....	121
<code>\phrasingSlurDashed</code> .....	136	<code>\semiflat</code> .....	721
<code>\phrasingSlurDashPattern</code> .....	136	<code>\semiGermanChords</code> .....	421
<code>\phrasingSlurDotted</code> .....	136	<code>\semisharp</code> .....	721
<code>\phrasingSlurDown</code> .....	136	<code>\sesquiflat</code> .....	721
<code>\phrasingSlurHalfDashed</code> .....	136	<code>\sesquisharp</code> .....	721
<code>\phrasingSlurHalfSolid</code> .....	136	<code>\set</code> .....	86, 601, 607
<code>\phrasingSlurNeutral</code> .....	136	<code>\sf</code> .....	124
<code>\phrasingSlurSolid</code> .....	136	<code>\sff</code> .....	124
<code>\phrasingSlurUp</code> .....	136	<code>\sfz</code> .....	124
<code>\phrygian</code> .....	22	<code>\shape</code> .....	633
<code>\pitchedTrill</code> .....	149	<code>\sharp</code> .....	721
<code>\portato</code> .....	121	<code>\shiftOff</code> .....	177
<code>\postscript</code> .....	253, 714	<code>\shiftOn</code> .....	177
<code>\powerChords</code> .....	385	<code>\shiftOnn</code> .....	177
<code>\pp</code> .....	124	<code>\shiftOnnn</code> .....	177
<code>\ppp</code> .....	124	<code>\shortfermata</code> .....	121
<code>\pppp</code> .....	124	<code>\showKeySignature</code> .....	400
<code>\ppppp</code> .....	124	<code>\showStaffSwitch</code> .....	330
<code>\prall</code> .....	121	<code>\signumcongruentiae</code> .....	121
<code>\pralldown</code> .....	121	<code>\simple</code> .....	689
<code>\prallmordent</code> .....	121	<code>\skip</code> .....	60, 286
<code>\prallprall</code> .....	121	<code>\slashed-digit</code> .....	735
<code>\prallup</code> .....	121	<code>\slashedGrace</code> .....	114
<code>\predefinedFretboardsOff</code> .....	380	<code>\slurDashed</code> .....	133
<code>\predefinedFretboardsOn</code> .....	380	<code>\slurDashPattern</code> .....	134
<code>\property-recursive</code> .....	734	<code>\slurDotted</code> .....	133
<code>\pt</code> .....	612	<code>\slurDown</code> .....	133
<code>\pushToTag</code> .....	508	<code>\slurHalfDashed</code> .....	133
<code>\put-adjacent</code> .....	703	<code>\slurHalfSolid</code> .....	133
<code>\quilisma</code> .....	446, 453	<code>\slurNeutral</code> .....	133
<code>\quoteDuring</code> .....	211, 214	<code>\slurSolid</code> .....	133
<code>\raise</code> .....	248, 703	<code>\slurUp</code> .....	134
<code>\relative</code> .....	2, 5, 14, 329	<code>\small</code> .....	220, 247, 689
<code>\RemoveEmptyStaves</code> .....	204, 205	<code>\smallCaps</code> .....	689
<code>\removeWithTag</code> .....	505	<code>\smaller</code> .....	245, 247, 690
<code>\repeat</code> .....	150	<code>\snappizzicato</code> .....	121

<code>\sostenutoOff</code> .....	332	<code>\treCorde</code> .....	332
<code>\sostenutoOn</code> .....	332	<code>\triangle</code> .....	253, 716
<code>\southernHarmonyHeads</code> .....	41	<code>\trill</code> .....	121, 148
<code>\southernHarmonyHeadsMinor</code> .....	41	<code>\tuplet</code> .....	48, 76
<code>\sp</code> .....	124	<code>\tupletDown</code> .....	48
<code>\spp</code> .....	124	<code>\tupletNeutral</code> .....	48
<code>\staccatissimo</code> .....	121	<code>\tupletUp</code> .....	48
<code>\staccato</code> .....	121	<code>\turn</code> .....	121
<code>\startGroup</code> .....	234	<code>\tweak</code> .....	605, 607
<code>\startStaff</code> .....	197, 200	<code>\type</code> .....	594
<code>\startTrillSpan</code> .....	148	<code>\typewriter</code> .....	692
<code>\stdBass</code> .....	726	<code>\unaCorda</code> .....	332
<code>\stdBassIV</code> .....	727	<code>\underline</code> .....	245, 692
<code>\stdBassV</code> .....	728	<code>\undertie</code> .....	692
<code>\stdBassVI</code> .....	729	<code>\unfoldRepeats</code> .....	521
<code>\stemDown</code> .....	229	<code>\unHideNotes</code> .....	226
<code>\stemNeutral</code> .....	229	<code>\unset</code> .....	602
<code>\stemUp</code> .....	229	<code>\upbow</code> .....	121, 336
<code>\stencil</code> .....	735	<code>\upmordent</code> .....	121
<code>\stopGroup</code> .....	234	<code>\upprall</code> .....	121
<code>\stopped</code> .....	121	<code>\upright</code> .....	693
<code>\stopStaff</code> .....	197, 200, 204	<code>\varcoda</code> .....	121
<code>\stopTrillSpan</code> .....	148	<code>\vcenter</code> .....	705
<code>\storePredefinedDiagram</code> .....	373	<code>\verbatim-file</code> .....	735
<code>\stringTuning</code> .....	356	<code>\verylongfermata</code> .....	121
<code>\strophæ</code> .....	446, 453	<code>\virga</code> .....	446, 453
<code>\strut</code> .....	735	<code>\virgula</code> .....	445
<code>\sub</code> .....	246, 690	<code>\voiceFourStyle</code> .....	176
<code>\super</code> .....	246, 690	<code>\voiceNeutralStyle</code> .....	176
<code>\sustainOff</code> .....	332	<code>\voiceOne</code> .....	173
<code>\sustainOn</code> .....	332	<code>\voiceOne ... \voiceFour</code> .....	173
<code>\tabChordRepeats</code> .....	343	<code>\voiceOneStyle</code> .....	176
<code>\tabFullNotation</code> .....	342	<code>\voiceThreeStyle</code> .....	176
<code>\table-of-contents</code> .....	502, 738	<code>\voiceTwoStyle</code> .....	176
<code>\tag</code> .....	505	<code>\void</code> .....	526
<code>\tagGroup</code> .....	508	<code>\vspace</code> .....	705
<code>\taor</code> .....	400	<code>\walkerHeads</code> .....	41
<code>\teeny</code> .....	220, 247, 690	<code>\walkerHeadsMinor</code> .....	41
<code>\tempo</code> .....	70	<code>\whiteout</code> .....	736
<code>\tenuto</code> .....	121	<code>\with</code> .....	587, 592
<code>\text</code> .....	691	<code>\with-color</code> .....	227, 736
<code>\textLengthOff</code> .....	63, 236	<code>\with-dimensions</code> .....	737
<code>\textLengthOn</code> .....	63, 236	<code>\with-link</code> .....	737
<code>\textSpannerDown</code> .....	237	<code>\with-url</code> .....	716
<code>\textSpannerNeutral</code> .....	237	<code>\woodwind-diagram</code> .....	725
<code>\textSpannerUp</code> .....	237	<code>\wordwrap</code> .....	250, 707
<code>\thumb</code> .....	121, 224	<code>\wordwrap-field</code> .....	706
<code>\thumb</code> .....	740	<code>\wordwrap-internal</code> .....	738
<code>\tie</code> .....	691	<code>\wordwrap-lines</code> .....	256, 738
<code>\tied-lyric</code> .....	721	<code>\wordwrap-string</code> .....	707
<code>\tieDashed</code> .....	55	<code>\wordwrap-string-internal</code> .....	738
<code>\tieDotted</code> .....	55		
<code>\tieDown</code> .....	55		
<code>\tieNeutral</code> .....	55	.....	111
<code>\tieSolid</code> .....	55		
<code>\tieUp</code> .....	55	~	
<code>\time</code> .....	65, 86	~.....	53
<code>\tiny</code> .....	220, 247, 691		
<code>\tocItem</code> .....	502	<b>1</b>	
<code>\translate</code> .....	249, 705	15ma.....	24
<code>\translate-scaled</code> .....	249, 705		
<code>\transparent</code> .....	735		
<code>\transpose</code> .....	5, 11, 14		
<code>\transposedCueDuring</code> .....	218		
<code>\transposition</code> .....	26, 211		

## 8

8va.....	24
8ve.....	24

## A

a capo, testo .....	250
abbellimenti .....	114, 121
abbellimenti al termine di una nota .....	115
abbellimenti e testo cantato .....	294
abbellimenti, modifica delle	
impostazioni di formattazione .....	116
abbellimenti, modifica manuale .....	116
<b>absolute</b> .....	781
accento .....	122, 740
accentus .....	741
<b>accepts</b> .....	594
acciaccatura .....	114
<b>acciaccatura</b> .....	781
acciaccatura su più note .....	118
Accidental, musica ficta .....	440
accidentals .....	440, 444, 455
<b>accidentalStyle</b> .....	781
accollatura .....	191
accordi .....	167
accordi e legature di valore .....	54
accordi e ottava relativa .....	5
accordi per chitarra, tabella .....	81
accordi vuoti .....	168
accordi, alterazioni in .....	34
accordi, altezza relativa .....	169
accordi, diteggiatura .....	225
accordion .....	333
accordion discant symbols .....	333
accordion shift symbols .....	333
accordion shifts .....	333
accordo, ripetizione .....	169
<b>addChordShape</b> .....	373, 781
adding a white background to text .....	736
adding custom fret diagrams .....	372
<b>addInstrumentDefinition</b> .....	781
<b>additionalPitchPrefix</b> .....	419
additions, in chords .....	414
<b>addQuote</b> .....	211, 781
adjusting staff symbol .....	613
<b>aeolian</b> .....	22
<b>afterGrace</b> .....	115, 781
<b>afterGraceFraction</b> .....	746
agogo .....	741
Aiken, testa di nota .....	41
<b>aikenHeads</b> .....	41
<b>aikenHeadsMinor</b> .....	41
aiuto, nuvoletta .....	231
al niente .....	128
<b>alias</b> .....	594
align to objects .....	629
<b>alignAboveContext</b> .....	597
<b>alignBelowContext</b> .....	284, 597
alist .....	744
allineamento orizzontale del testo .....	248
allineamento sulla cadenza .....	119
allineamento verticale del testo .....	248
allineamento, testo, comandi .....	251
allineare il markup .....	248

allineare il testo .....	248
<b>allowPageTurn</b> .....	781
<b>allowVoltaHook</b> .....	781
alterazione .....	5
alterazione di cortesia .....	6
alterazione di sicurezza .....	6
alterazione di un quarto di tono .....	8
alterazione e legatura di valore .....	6
alterazione tra parentesi .....	6
alterazione, di cortesia .....	6
alterazione, di sicurezza .....	6
alterazione, quarto di tono .....	8
alterazione, tra parentesi .....	6
alterazioni .....	28
alterazioni automatiche .....	28
alterazioni di precauzione in stile moderno .....	30
alterazioni e note simultanee .....	34
alterazioni in stile moderno .....	30
alterazioni moderne .....	31
alterazioni negli accordi .....	34
alterazioni su più voci .....	31
alterazioni, cadenze .....	75
alterazioni, musica in tempo libero .....	75
alterazioni, stile <i>modern-cautionary</i> .....	30
alterazioni, stile moderno delle .....	30
<b>alterBroken</b> .....	781
altered chords .....	413
altezza naturale .....	6
altezza relativa, accordi .....	169
altezze .....	1
altezze, trasposizione delle .....	11
alto, chiave di .....	17
Amazing Grace bagpipe example .....	401
ambito delle altezze .....	35
ambitus .....	35
anacrusi .....	73
anacrusi in una ripetizione .....	152
analisi musicologica .....	234
analizzatore sintattico .....	746
angled hairpins .....	626
annidamento dei rigli .....	194
annidamento, ripetizioni .....	158
<b>annotate-spacing</b> .....	576
annotazione .....	243
annotazione su pausa multipla .....	62
anthems .....	300
antica, chiave .....	17
aperto .....	741
aperto (\open) .....	121
apice .....	246
<b>appendToTag</b> .....	781
<b>applyContext</b> .....	781
<b>applyMusic</b> .....	781
<b>applyOutput</b> .....	781
appoggiatura .....	114
<b>appoggiatura</b> .....	781
Arabic key signatures .....	466
Arabic music .....	464
Arabic music example .....	468
Arabic music template .....	468
Arabic note names .....	465
Arabic semi-flat symbol .....	465
Arabic time signatures .....	467
arcata in giù .....	121
arcata in su .....	121

arcate, in su e in giù.....	741
armatura di chiave.....	5, 22
armonico (\flageolet).....	121
armonico, testa di nota.....	38
arpeggio.....	144
arpeggio attraverso il rigo, stile della parentesi ...	147
arpeggio spezzato .....	144
arpeggio, simboli speciali.....	145
arpeggioArrowDown .....	144
arpeggioArrowUp.....	144
arpeggioBracket.....	145
arpeggioNormal.....	144
arpeggioParenthesis.....	145
arpeggioParenthesisDashed.....	145
arrow-head .....	253
articolazione "espressivo".....	126
articolazioni.....	121
articulate script .....	525
articulate.ly .....	525
articulation-event .....	213
articulations.....	445
artificial harmonics.....	337
assertBeamQuant.....	782
assertBeamSlope.....	782
associatedVoice.....	263
associatedVoice .....	265, 295
assoluta, ottava.....	1
assoluto .....	1
audio .....	514
aug .....	412
auto-first-page-number.....	538
autoBeaming.....	86, 579
autoBeamOff .....	83
autoBeamOn .....	83
autochange.....	328, 782
autochange and relative music .....	329
automatic chord diagrams.....	379
automatic fret diagrams.....	379
automatic staff changes.....	328
automaticBars .....	624

## B

Bézier curves, control points.....	632
backslashed digits .....	730
bagpipe .....	400
bagpipe example .....	401
Balloon_engraver .....	231
balloonGrobText.....	231, 782
balloonLengthOff .....	231
balloonLengthOn.....	231
balloonText .....	231, 782
banjo tablature.....	339
banjo tablatures.....	387
banjo tunings .....	388
banjo-c-tuning.....	388
banjo-modal-tuning.....	388
banjo-open-d-tuning.....	388
banjo-open-dm-tuning.....	388
bar.....	99, 105, 782
bar lines, suppressing.....	624
barCheckSynchronize.....	111
baritono, chiave di.....	17
BarNumber .....	106
barNumberCheck.....	111, 782

barNumberVisibility.....	106
barrata, testa di nota.....	38
barre indications .....	360
Bartók pizzicato.....	338
bartype.....	105
base-shortest-duration.....	565
baseMoment .....	86
bass note, for chords .....	415
Bass, figured .....	424
Bass, thorough.....	424
basso.....	741
basso acustico.....	741
Basso continuo.....	424
basso, chiave di.....	17
battiti per minuto.....	70
battuta in levare .....	73
battuta, controlli.....	111
battuta, numeri .....	106
battuta, numero, formato del .....	108
battuta, stanghette.....	99
battuta, stanghette manuali .....	99
battute dei ritornelli.....	99
beamExceptions .....	86, 782
beams, cross-staff .....	326
beatStructure .....	86
bemolle .....	6
bemolle, doppio .....	6
bendAfter.....	139, 782
bilanciamento nel MIDI .....	524
binari ferroviari .....	138
binding-offset.....	535
bisbiglando.....	334
Bison .....	746
blank-after-score-page-penalty .....	537
blank-last-page-penalty .....	537
blank-page-penalty.....	537
BNF .....	746
bold.....	245
bongo.....	741
bookOutputName.....	782
bookOutputSuffix .....	782
bookTitleMarkup.....	485
bottom-margin.....	531
bounding box .....	613
bowing indications .....	336
box .....	251
bracket .....	130, 251, 332
break-align-symbols .....	629
break-visibility .....	620
breakable.....	84
breakbefore .....	482
breaking lines .....	543
breathe.....	137, 782
breve.....	45, 57
broken spanners, modifying.....	636



## C

cabasa .....	741	chord quality .....	412
cadenza .....	75, 119	chord shapes for fretted instruments .....	373
cadenza, allineamento su .....	119	chord steps, altering .....	415
cadenza, alterazioni .....	75	chord, modifying one note in .....	606
cadenza, interruzioni di linea .....	76	chord, repetition .....	343
cadenza, interruzioni di pagina .....	76	<b>chordChanges</b> .....	417
cadenza, numeri di battuta .....	75	<b>chordmode</b> .....	5, 14, 370
cadenza, stanghette .....	75	<b>chordNameExceptions</b> .....	420
cadenza, travature .....	75	<b>chordNameLowercaseMinor</b> .....	419
<b>cadenzaOff</b> .....	75	<b>ChordNames</b> .....	370
<b>cadenzaOn</b> .....	75	<b>chordNameSeparator</b> .....	420
callback .....	744	<b>chordNoteNamer</b> .....	420
cambiare i tipi di carattere .....	245	<b>chordPrefixSpacer</b> .....	421
canali MIDI .....	521	<b>chordRepeats</b> .....	782
cantante, nome del .....	293	<b>chordRootNamer</b> .....	419
canti .....	312	chords .....	416
capo .....	364	chords, cross-staff .....	330
carattere tipografico, famiglia di .....	745	chords, jazz .....	418
caratteri non-ASCII .....	509	chords, power .....	385
caratteri riservati, stampare .....	244	chords, splitting across staves with \autochange .....	329
caratteri speciali .....	509	chords, suppressing repeated .....	417
caratteri speciali in modalità markup .....	244	Christian Harmony, testa di nota .....	41
carta, formato .....	529	<b>circle</b> .....	251
<b>center-align</b> .....	248	circling text .....	709
<b>center-column</b> .....	249	circulus .....	741
centered dynamics in piano music .....	325	citare le voci .....	211
centering a column of text .....	693	citazioni in corpo più piccolo, togliere le .....	218
centrare il testo sulla pagina .....	250	citazioni in corpo piccolo, chiavi .....	17
cesura .....	138	citazioni nel testo vocale .....	262
<b>change</b> .....	326	claves .....	741
changing direction of text columns .....	694	<b>clef</b> .....	17, 782
changing properties .....	601	clef, moderntab .....	358
changing staff automatically .....	328	clef, percussion .....	389
changing staff manually .....	326	clef, tab .....	358
<b>check-consistency</b> .....	534	clef, visibility following explicit change .....	622
chiave .....	5, 17	clefs .....	436, 443, 454
chiave antica .....	17	clefs, visibility of transposition .....	624
chiave di baritono .....	17	<b>clip-regions</b> .....	511
chiave di basso .....	17	closure .....	744
chiave di contralto .....	17	cluster .....	172
chiave di Do .....	17	cluster di note .....	172
chiave di Fa .....	17	coda .....	113, 121, 741
chiave di mezzosoprano .....	17	coda sulla stanghetta .....	239
chiave di Sol .....	17	collisione, numeri di battuta .....	110
chiave di soprano .....	17	collisioni .....	177
chiave di subbasso .....	17	collisioni di note .....	177
chiave di tenore .....	17	collisioni, ignorare .....	172
chiave di tenore per coro .....	17	collisioni, ignorare .....	181
chiave di varbaritono .....	17	collisions, cross-staff voices .....	326
chiave di violino .....	17	colonne, testo .....	249
chiave francese .....	17	<b>color</b> .....	227
chiave traspositrice .....	17	colorare gli oggetti .....	227
chiavi con notine (citazioni in corpo piccolo) .....	17	colorare le note .....	227
chitarra, teste di nota .....	38	colorare le voci .....	176
chiuso (\stopped) .....	121	colorate, note .....	227
chord chords .....	411	colorati, oggetti .....	227
chord diagrams .....	359, 369	colore negli accordi .....	228
chord diagrams, automatic .....	379	colore rgb .....	228
chord glissandi .....	353	colore x11 .....	228
chord inversions .....	415	colori .....	227
chord mode .....	411	colori, elenco dei .....	659
chord names .....	411, 416	coloring objects .....	620
chord names with fret diagrams .....	370	coloring text .....	736
		<b>column</b> .....	249

combinare musica contrassegnata con etichette...	508
combinatore delle parti	181
combinazione automatica delle parti	181
<b>combine</b>	253
comma intervals	469
<b>common-shortest-duration</b>	565
<b>Completion_heads_engraver</b>	79
<b>Completion_rest_engraver</b>	79
<b>compoundMeter</b>	782
<b>compressMMRests</b>	61, 63, 782
comprimere la musica	53
concatenating text	694
condensare le pause normali	65
conga	741
<b>consists</b>	594
costante-hairpins	128
context properties, changing defaults	589
Contexts, creating and referencing	581
contexts, defining new	594
contexts, implicit	597
contexts, keeping alive	584
contexts, layout order	596
contexts, lifetime	584
control points, Bézier curves	632
control points, tweaking	607
controlli del numero di battuta	111
controlli di battuta	111
controlli di battuta con ripetizioni	152
controlling general text alignment	696
controllo dell'ottava	10
controllo della misura	111
controllo delle altezze	10
<b>controlpitch</b>	10
copyright, segno	510
coro, rigo per	191
corona	113, 121, 740
corona sulla stanghetta	239
correzione dell'ottava	10
cowbell	741
<b>cr</b>	125
creating empty text objects	733
creating horizontal spaces in text	698
creating text fractions	731
creating vertical spaces in text	705, 735
creazione del rigo	190
<b>cresc</b>	126
crescendo	125
crescendo-event	213
<b>crescHairpin</b>	126
<b>crescTextCresc</b>	126
<b>cross</b>	38
cross staff chords	330
cross staff line	330
cross staff notes	330
cross staff stems	330
cross-staff	330
cross-staff beams	326
cross-staff chords	330
cross-staff collisions	326
cross-staff line	330
cross-staff notes	326, 330
cross-staff stems	330
<b>crossStaff</b>	783
<b>cueClef</b>	214, 783
<b>cueClefUnset</b>	783

<b>cueDuring</b>	214, 783
<b>cueDuringWithClef</b>	214, 783
cues	306
CueVoice	214
cuica	741
<b>currentBarNumber</b>	106, 120
custodes	434
custom fret diagrams	359
custom fret diagrams, adding	372
custom string tunings	356
customized fret diagram	366
customizing chord names	418
custos	434

## D

D.S. al Fine	113
dampened notes on fretted instruments	384
<b>deadNote</b>	783
decorazione del testo	251
<b>decr</b>	125
<b>decresc</b>	126
decrescendo	125
<b>default</b>	28, 29
default context properties, changing	589
<b>default-staff-staff-spacing</b>	550
<b>defaultBarType</b>	105
<b>defaultNoteHeads</b>	783
<b>defaultTimeSignature</b>	66
<b>defineBarLine</b>	103, 783
definire le stanghetta	103
definizioni di contesto per il MIDI	519
delimitatori di inizio del sistema	191
delimitatori di inizio del sistema annidati	194
<b>denies</b>	594
diagram, fret, customized	366
diagramma degli accordi per chitarra	81
diagrams, chord for fretted instruments	359
diagrams, fret	359
diagrams, fret, transposing	371
diamante, testa di nota	38
diamond-shaped note heads	337
diesis	6
diesis, doppio	6
<b>dim</b>	126, 412
dimensione del testo	245
dimensione del tipo di carattere	245
dimensione del tipo di carattere (elementi della notazione)	220
dimensioni del tipo di carattere standard (per gli elementi della notazione)	223
dimensions	613
<b>dimHairpin</b>	126
diminuendo	125
<b>dimTextDecr</b>	126
<b>dimTextDecresc</b>	126
<b>dimTextDim</b>	126
dinamica, nuovi segni di	130
dinamiche	124
dinamiche assolute	124
dinamiche editoriali	130
dinamiche nel MIDI	516
dinamiche, parentesi	130
dinamiche, posizionamento verticale	127

diritto d'autore.....	510
discant symbols, accordion.....	333
disegnare oggetti grafici .....	251
<b>displayLilyMusic</b> .....	783
<b>displayMusic</b> .....	783
<b>displayScheme</b> .....	783
disposizione delle travature, proprietà predefinite delle indicazioni di tempo .....	66
distances, absolute .....	612
distances, scaled.....	612
distanza tra i righi .....	549
dita, cambio.....	224
diteggiatura .....	224
diteggiatura per accordi .....	225
diteggiature e pause multiple .....	65
divisio .....	444
divisione delle note .....	79
divisione delle pause .....	79
divisiones .....	444
Do, chiave di .....	17
<i>dodecafonico</i> , stile delle alterazioni.....	32
dodecafonico, stile delle alterazioni.....	33
dodecafonico, stile neomoderno .....	33
<b>dodecaphonic</b> .....	32
<b>dodecaphonic-first</b> .....	33
<b>dodecaphonic-no-repeat</b> .....	33
doppio bemolle .....	6
doppio diesis .....	6
doppio mordente (\prallmordent).....	121
doppio mordente (\prallprall).....	121
doppio punto, note .....	46
<b>dorian</b> .....	22
dorico.....	22
<b>dotsDown</b> .....	46
<b>dotsNeutral</b> .....	46
<b>dotsUp</b> .....	46
down bow indication .....	336
downbow.....	741
downmordent .....	740
downprall.....	740
<b>draw-circle</b> .....	253
<b>draw-line</b> .....	253
drawing a line across a page.....	710
drawing beams within text .....	708
drawing boxes with rounded corners .....	712
drawing boxes with rounded corners around text .....	715
drawing circles within text.....	709
drawing dashed lines within text.....	709
drawing dotted lines within text.....	710
drawing ellipse around text .....	711
drawing lines within text .....	710
drawing oval around text .....	713
drawing paths .....	714
drawing solid boxes within text .....	712
drawing squiggled lines within text .....	711
drawing staff symbol .....	613
drawing triangles within text .....	716
<b>drummode</b> .....	190
drums.....	388, 390
<b>DrumStaff</b> .....	190
durata delle note .....	45
durata predefinita .....	46
durate, scalare.....	52
<b>dynamic</b> .....	130

dynamic-event.....	213
<b>dynamicDown</b> .....	127
<b>DynamicLineSpanner</b> .....	127
<b>dynamicNeutral</b> .....	127
dynamics, centered in keyboard music .....	325
<b>dynamicUp</b> .....	127

## E

<b>easyHeadsOff</b> .....	39
<b>easyHeadsOn</b> .....	39
editoriali, dinamiche .....	130
effetti nel MIDI .....	524
elementi testuali non vuoti .....	236
elenare i tipi di carattere disponibili .....	259
elenco dei colori .....	659
encapsulated postscript, output.....	513
enclosing text in a box with rounded corners.....	715
enclosing text within a box .....	684
<b>endSpanners</b> .....	783
engravers, including in contexts.....	594
eolio .....	22
EPS, output .....	513
<b>epsfile</b> .....	253
equalizzazione MIDI.....	516
espandere la musica .....	53
espressione nel MIDI .....	524
espressioni di markup .....	243
espressivo .....	121
<b>espressivo</b> .....	126
espressivo .....	740
estensione .....	35
estensore.....	273
estensori del testo .....	237
estensori del testo, formattazione .....	237
etichetta .....	505
etichette, gruppi di .....	508
<b>eventChords</b> .....	783
eventi segnaposto.....	168
exceptions, chord names.....	421
<b>explicitClefVisibility</b> .....	622
<b>explicitKeySignatureVisibility</b> .....	622
extended chords .....	413
<b>extra-offset</b> .....	550

## F

<b>f</b> .....	124
Fa, chiave di.....	17
famiglie di tipi di carattere .....	246
famiglie di tipi di carattere, impostare .....	259
<b>featherDurations</b> .....	98, 783
fermata.....	740
fermata su pausa multipla .....	62
<b>fermataMarkup</b> .....	62, 63
Ferneyhough, forcelle.....	128
Feta, tipo di carattere .....	660
<b>ff</b> .....	124
<b>fff</b> .....	124
<b>ffff</b> .....	124
<b>fffff</b> .....	124
Figured bass .....	424
figured bass alignment.....	429
figured bass extender lines.....	427
<b>fill-line</b> .....	250

<b>filled-box</b> .....	253
finali alternati, ripetizioni .....	160
finali alternativi .....	150
finali alternativi con legature di valore .....	153
finali alternativi e testo vocale .....	285
finalis .....	444
fine ripetizione .....	158
<b>finger</b> .....	224, 783
fingering vs. string numbers .....	339
fingerings, adding to fret diagrams .....	381
fingerings, right hand for fretted instruments .....	382
<b>first-page-number</b> .....	538
<b>fixed</b> .....	2, 783
flageolet .....	741
flags .....	439
flared-hairpin .....	128
Flex .....	746
floor tom tom .....	741
follow voice .....	330
<b>followVoice</b> .....	330
font .....	257, 745
font non testuali nel markup .....	257
font, cambiare .....	245
<b>font-interface</b> .....	223, 257
<b>font-size</b> .....	220, 223
<b>fontsize</b> .....	245
<b>fontSize</b> .....	220
<b>footnote</b> .....	783
forcella .....	125
forcelle allargate (flared-hairpins) .....	128
forcelle continue (costante-hairpins) .....	128
forcelle Ferneyhough .....	128
<b>forget</b> .....	34
<i>forget</i> , stile delle alterazioni .....	34
formato carta .....	529
formato carta, orientamento .....	530
formato carta, orizzontale (landscape) .....	530
formato del segno di chiamata .....	113
formato pagina .....	529
formattare gli estensori del testo .....	237
formattare le notine .....	214
formattazione del gruppo irregolare .....	49
formattazione della pagina .....	569
formattazione della terzina .....	49
formattazione mensurale .....	194
formattazione nel testo vocale .....	262
formattazione, oggetti .....	745
<b>four-string-banjo</b> .....	388
<b>fp</b> .....	124
frammenti .....	214
frammenti musicali .....	511
frammenti, citare i .....	211
francese, chiave .....	17
frase, legature di .....	136
fraseggio, nel testo vocale .....	270
fret .....	343
fret diagram, customized .....	366
fret diagrams .....	359, 369
fret diagrams with chord names .....	370
fret diagrams, adding custom .....	372
fret diagrams, adding fingerings .....	381
fret diagrams, automatic .....	379
fret diagrams, custom .....	359
fret diagrams, mandolin .....	369
fret diagrams, transposing .....	371

fret diagrams, ukulele .....	369
<b>fret-diagram</b> .....	360
fret-diagram markup .....	360
<b>fret-diagram-interface</b> .....	366
<b>fret-diagram-terse</b> .....	362
fret-diagram-terse markup .....	362
<b>fret-diagram-verbose</b> .....	364
fret-diagram-verbose markup .....	364
<b>FretBoards</b> .....	369
fretted instruments, chord shapes .....	373
fretted instruments, dampened notes .....	384
fretted instruments, harmonics .....	384
fretted instruments, indicating position and barring .....	384
fretted instruments, predefined string tunings .....	356
fretted instruments, right hand fingerings .....	382
frigio .....	22
Funk, testa di nota .....	41
<b>funkHeads</b> .....	41
<b>funkHeadsMinor</b> .....	41

## G

gambo .....	229
gambo barrato .....	116
gambo invisibile .....	229
gambo, direzione .....	229
gambo, giù .....	229
gambo, neutrale .....	229
gambo, su .....	229
<b>general-align</b> .....	249
gestione del tempo .....	120
ghost notes .....	229
giustificato, testo .....	250
glifi musicali .....	113
glifo .....	745
glissandi e ripetizioni .....	158
<b>glissando</b> .....	140
glyph .....	745
<b>grace</b> .....	784
grace notes .....	400
graffa verticale .....	191
graffe, varie dimensioni .....	257
grafica inclusa .....	253
grafica, inclusione .....	251
grammatica di LilyPond .....	746
Gregorian square neumes ligatures .....	446
<b>GregorianTranscriptionStaff</b> .....	190
<b>Grid_line_span_engraver</b> .....	232
<b>Grid_point_engraver</b> .....	232
<b>gridInterval</b> .....	232
griglie .....	232
grob .....	599, 745
grob properties .....	603
grob-interface .....	745
<b>grobdescriptions</b> .....	784
grobs, overwriting .....	620
grobs, visibility of .....	618
<b>grow-direction</b> .....	98
gruppetto (\turn) .....	121
gruppetto rovesciato (\reverseturn) .....	121
gruppi irregolari .....	48
gruppi irregolari, raggruppamento .....	48
gruppo di righe .....	191
gruppo irregolare, formattazione del .....	49

gruppo irregolare, modifiche del numero del .....	49
gruppo irregolare, posizionamento della	
parentesi quadra .....	48
Guida al funzionamento interno .....	579
guiro .....	741
guitar tablature .....	339

## H

hairpins, angled .....	626
half-open high hat .....	741
halfopen .....	741
<b>halign</b> .....	248
hammer on .....	354
handclap .....	741
harmonic indications in tablature notation .....	346
Harmonica Sacra, testa di nota .....	41
<b>harmonicByFret</b> .....	784
<b>harmonicByRatio</b> .....	784
<b>harmonicNote</b> .....	784
harmonics on fretted instruments .....	384
harmonics, artificial .....	337
harmonics, natural .....	337
<b>harmonicsOn</b> .....	784
harp pedal diagrams .....	334
harp pedals .....	334
harps .....	334
<b>hbracket</b> .....	251
<b>hide</b> .....	784
<b>hideKeySignature</b> .....	400
<b>hideNotes</b> .....	226
<b>hideStaffSwitch</b> .....	330
high bongo .....	741
high conga .....	741
high hat .....	741
high timbale .....	741
horizontal spacing, overriding .....	638
<b>horizontal-shift</b> .....	536
<b>Horizontal_bracket_engraver</b> .....	234
horizontally centering text .....	693
hufnagel .....	432, 433
<b>huge</b> .....	220, 247
hyphens .....	273

## I

ictus .....	741
immagini incluse .....	253
immutabili, oggetti .....	745
immutable .....	745
implicit contexts .....	597
importing stencils into text .....	735
<b>improvisationOff</b> .....	44, 81
<b>improvisationOn</b> .....	44, 81
improvvisazione .....	44
<b>incipit</b> .....	457, 784
incipits, adding .....	457
include-settings .....	509
inclusione di file .....	502
incorniciatura del testo .....	251
<b>indent</b> .....	208, 536, 569
indicating No Chord in ChordNames .....	417
indicating position and barring for	
fretted instruments .....	384

indicazione di tempo .....	65
indicazione di tempo, impostazioni predefinite .....	66
indicazione di tempo, stile .....	66
indicazione di tempo, visibilità dell' .....	66
indicazione manuale di ripetizione .....	158
indicazione metronomica .....	70
indicazione metronomica con testo .....	70
indicazioni di diteggiatura per accordi .....	225
indicazioni di tempo composto .....	78
indicazioni di tempo doppie .....	76
indicazioni di tempo polimetrico .....	76
indicazioni di tempo, ripristinare i valori	
predefiniti delle proprietà delle .....	68
indicazioni dinamiche multiple su una nota .....	125
indicazioni dinamiche nuove .....	130
indicazioni dinamiche, più di un	
segno su una nota .....	125
indicazioni polimetriche .....	76
indicazioni testuali .....	239
informazioni sul tempo e ripetizioni .....	158
ingrandimento del tipo di carattere .....	220
<b>inherit-acceptability</b> .....	784
inizializzazione del rigo .....	190
inizio del sistema .....	191
inizio ripetizione .....	158
inlining an Encapsulated PostScript image .....	712
<b>inner-margin</b> .....	535
inni .....	312
insegnamento ( <i>teaching</i> ), stile delle alterazioni .....	33
inserimento del testo .....	262
inserting music into text .....	720
inserting PostScript directly into text .....	714
inserting URL links into text .....	716
<b>inStaffSegno</b> .....	784
<b>instrumentSwitch</b> .....	784
intavolatura .....	190
interfacce dei grob .....	745
interfaccia .....	745
interface, layout .....	599
Internals Reference .....	579
interruzioni di linea .....	99
interruzioni di linea regolari .....	545
interruzioni di linea, cadenze .....	76
interruzioni di linea, musica in tempo libero .....	76
interruzioni di linea, travature .....	84
interruzioni di pagina .....	569
interruzioni di pagina, cadenze .....	76
interruzioni di pagina, musica in tempo libero .....	76
interruzioni nella musica in tempo libero .....	76
intestazioni .....	477
<b>inversion</b> .....	784
inversione .....	14
inversione modale .....	16
invisibile, gambo .....	229
invisibili, note .....	226
<b>ionian</b> .....	22
ionio .....	22
<b>italic</b> .....	245

## J

jazz chords .....	418
justified-lines .....	256
justify .....	250
justifying lines of text .....	737
justifying text .....	700

## K

keepWithTag .....	785
key .....	22, 41, 785
key signature .....	440, 444
key signature, visibility following	
explicit change .....	622
keyboard instrument staves .....	325
keyboard music, centering dynamics .....	325
keyed instrument staves .....	325
KievanStaff .....	453
KievanVoice .....	453
killCues .....	218, 785
kirchenpausen .....	63

## L

label .....	785
laissez vibrer .....	55
laissezVibrer .....	55
language .....	785
languageRestore .....	785
languageSaveAndChange .....	785
large .....	220, 247
larger .....	245, 247
last-bottom-spacing .....	533
layers .....	620
layout file .....	541
layout interface .....	599
layout-set-staff-size .....	541
left aligning text .....	701
left-align .....	248
left-margin .....	534
legatura di frase .....	133
legatura di frase puntata .....	136
legatura di frase tratteggiata .....	136
legatura di frase, definizione dei	
modelli di tratterggio .....	136
legatura di frase, metà continua e	
metà tratteggiata .....	136
legatura di portamento continua .....	133
legatura di portamento e ripetizioni .....	158
legatura di portamento punteggiata .....	133
legatura di portamento tratteggiata .....	133
legatura di portamento, definizione dei modelli di	
tratterggio per il fraseggio .....	136
legatura di portamento, definizione del	
modello di tratterggio .....	134
legatura di portamento, frase puntata .....	136
legatura di portamento, frase tratteggiata .....	136
legatura di portamento, fraseggio multiplo .....	136
legatura di portamento, fraseggio simultaneo .....	136
legatura di portamento, fraseggio, definizione dei	
modelli di tratterggio .....	136
legatura di portamento, metà	
tratteggiata e metà continua .....	133

legatura di portamento, tratto metà continuo	
e metà tratteggiato .....	136
legatura di valore .....	53
legatura di valore e alterazione .....	6
legatura di valore, laissez vibrer .....	55
legature di frase .....	136
legature di frase multiple .....	136
legature di frase simultanee .....	136
legature di portamento .....	133
legature di portamento multiple .....	133
legature di portamento simultanee .....	133
legature di portamento,	
posizionamento manuale .....	133
legature di portamento, sopra le note .....	133
legature di portamento, sotto le note .....	133
legature di portamento, stile .....	133
legature di valore e accordi .....	54
legature di valore e parentesi della volta .....	55
legature di valore punteggiate .....	55
legature di valore tratteggiate .....	55
legature di valore, aspetto .....	55
legature di valore, finali alternativi .....	153
legature di valore, nelle ripetizioni .....	153
legature di valore, posizionamento .....	55
legature di valore, ripetizione .....	55
legature, nel testo vocale .....	269
lexer .....	746
lheel .....	741
lidio .....	22
Ligatures .....	434, 456
ligatures in text .....	694
line, cross-staff .....	330
line, staff-change .....	330
line, staff-change follower .....	330
line-width .....	534, 569
linee del rigo, fermare e avviare .....	197
linee del rigo, modificare .....	197
linee verticali tra i rigi .....	232
lineprall .....	740
lingua, nomi delle altezze in un'altra .....	8
lingua, nomi delle note in un'altra .....	8
lista di associazioni .....	744
livello del chorus nel MIDI .....	524
locrian .....	22
locrio .....	22
longa .....	45, 57
longfermata .....	740
low bongo .....	741
low conga .....	741
low timbale .....	741
lower .....	248
lowering text .....	702
ltoe .....	741
lunghezza delle note .....	45
ly:minimal-breaking .....	548
ly:one-line-breaking .....	548
ly:optimal-breaking .....	548
ly:page-turn-breaking .....	548
lydian .....	22
lyrics, aligning with sporadic melody .....	586

## M

m	412
maggiore	22
magnification->font-size	220, 541
magnify	245
magnifying text	686
magnifyMusic	220, 785
magnifyStaff	785
magstep	220, 541, 612
maj	412
major	22
major seven symbols	421
majorSevenSymbol	419
makam	469
makamlar	469
make-dynamic-script	131
make-pango-font-tree	259
makeClusters	172, 785
makeDefaultStringTuning	785
manual line breaks	543
manual staff changes	326
Manuali	1
maqam	464
maqams	464
maracas	741
marcato	122, 740
margin di rilegatura	535
margin, testo che va oltre	237
mark	112, 239, 785
markLengthOff	71, 240
markLengthOn	71, 240
markup	239, 241, 242, 243
markup condizionale	488
markup multilinea	249
markup, allineare	248
markup, centrare sulla pagina	250
markup, comandi di allineamento del testo	251
markup, decorazione	251
markup, espressioni	243
markup, incorniciatura	251
markup, multipagina	256
markup, notazione musicale dentro	254
markup, padding	252
markup, partitura dentro	255
markup, sintassi	243
markup, testo a capo	250
markup, testo giustificato	250
markup-markup-spacing	533
markup-system-spacing	533
markuplist	242, 256, 257
markupMap	785
max-systems-per-page	536
maxima	45, 57
measureLength	86, 120
measurePosition	73, 120
Medicaea, Editio	432, 433
medium intervals	464
melisma	270, 273
melismi, con travature	84
melodia alternativa, passare a	295
melodia, mostrare i ritmi della	81
mensural	432, 433
Mensural ligatures	441
mensurale, formattazione	194
MensuralStaff	190
MensuralStaff	435
MensuralVoice	435
mensuration sign	437
mergeDifferentlyDottedOff	177
mergeDifferentlyDottedOn	177
mergeDifferentlyHeadedOff	177
mergeDifferentlyHeadedOn	177
merging text	694, 702
mezzosoprano, chiave di	17
mf	124
micro-tones, tab	358
microtoni	9
MIDI	26, 514
MIDI block	515
MIDI using repeats	521
MIDI, canali	521
MIDI, definizioni di contesto	519
MIDI, dinamiche	516
MIDI, equalizzazione	516
MIDI, notazione non supportata	514
MIDI, notazione supportata	514
MIDI, strumenti	525
MIDI, tracce	521
MIDI, volume	516
midiBalance	524
midiChannelMapping	521
midiChorusLevel	524
midiExpression	524
midiPanPosition	524
midiReverbLevel	524
min-systems-per-page	536
minimum-Y-extent	550
minimumFret	343, 381
minimumPageTurnLength	548
minimumRepeatLengthForPageTurn	549
minor	22
minorChordModifier	420
minore	22
mirroring markup	715
misolidio	22
misura	65
misura parziale	73
misura, numeri	106
misura, raggruppamenti	92
misura, ripetizioni	162
misura, sottoraggruppamenti	92
misura, stanghette	99
misura, stanghette manuali	99
mixed	332
mixolydian	22
modale, inversione	16
modale, trasposizione	15
modali, trasposizioni	15
modalInversion	16, 785
modalità markup, caratteri speciali	244
modalità markup, testo tra virgolette	244
modalTranspose	15, 785
mode	746
modern	30
modern, stile delle alterazioni	30, 31
modern-cautionary	30
modern-cautionary, stile delle alterazioni	30
modern-voice	31
modern-voice-cautionary	31

<i>modern-voice-cautionary</i> , stile delle alterazioni	31
moderntab clef	358
modi	22
modi ecclesiastici	22
modificare gli abbellimenti	116
modificare i nomi degli strumenti	209
modifiers, in chords	412
mordent	740
mordente inferiore (\mordent)	121
mordente inferiore, giù	121
mordente inferiore, su	121
mordente superiore (\prall)	121
mordente superiore, giù	121
mordente superiore, su	121
movimenti, molteplici	472
mp	124
multilinea, markup	249
multilinea, testo	249
MultiMeasureRestText	62
multipagina, testo	256
musica a quattro battute	545
musica dentro il blocco markup	254
Musica ficta	440
musica in tempo libero	75, 120
musica in tempo libero, alterazioni	75
musica in tempo libero, interruzioni di linea	76
musica in tempo libero, interruzioni di pagina	76
musica in tempo libero, numeri di battuta	75
musica in tempo libero, stanghette	75
musica in tempo libero, travature	75
musica mensurale, trascrizione di	194
musica parallela	187
musica per principianti	39
musica polifonica	177
musica religiosa	312
musica rinascimentale	194
musicglyph	113
musicMap	786
musicologia, analisi	234
musicQuotes	746
mutable	746
mute bongo	741
mute conga	741
mute timbale	741

## N

N.C. symbol	417
name	594
nascondere i righi	204
nascondere i righi antichi	205
nascondere i righi ritmici	205
nascoste, note	226
natural harmonics	337
neo-modern	32
neo-modern, stile delle alterazioni	32
neo-modern-cautionary	32
neo-modern-cautionary, stile delle alterazioni	32
neo-modern-voice	32
neo-modern-voice, stile delle alterazioni	32
neo-modern-voice-cautionary	32
neo-modern-voice-cautionary, stile	
delle alterazioni	32
neomensural	433
new contexts	581

niente, al	128
no chord symbol	417
no-reset	33
no-reset, stile delle alterazioni	33
noBeam	95
nome del cantante	293
nomi degli strumenti	207, 525
nomi degli strumenti abbreviati	207
nomi degli strumenti, aggiungerli	
ad altri contesti	209
nomi degli strumenti, centrare	208
nomi degli strumenti, complessi	208
nomi degli strumenti, modifica	209
nomi dei personaggi	304
nomi delle altezze	1
nomi delle altezze, altre lingue	8
nomi delle note predefiniti	6
nomi delle note, altre lingue	8
nomi delle note, olandese	6
nomi delle note, predefinito	6
nonstaff-nonstaff-spacing	550
nonstaff-relatedstaff-spacing	550
nonstaff-unrelatedstaff-spacing	550
noPageBreak	786
noPageTurn	786
normal-size-super	246
normalsize	220, 247
nota spaziatrice	60
nota, durata predefinita	46
nota, spostamento	177
notazione dentro il blocco markup	254
notazione grafica	253
notazione semplificata	39
notazione, dimensione del tipo di carattere	220
notazione, spiegare la	231
note a forma variabile	41
note a piè di pagina	490
note a piè di pagina basate su un evento	491
note a piè di pagina basate sul tempo	493
note a piè di pagina nel testo separato	496
note a piè di pagina nelle espressioni musicali	490
note colorate	227
note colorate negli accordi	228
note doppiamente puntate	46
note fantasma	229
note heads, ancient	438, 454
note heads, diamond-shaped	337
note in corpo più piccolo	211, 214
note invisibili	226
note nascoste	226
note più piccole	214
note puntate	46
note simultanee e alterazioni	34
note tra parentesi	229
note trasparenti	226
note, divisione	79
note, durata delle	45
note, lunghezza delle	45
note, spaziatura orizzontale	566
note, trasposizione delle	11
note-event	213
Note_heads_engraver	79
notes within text by log and dot-count	718
notes within text by string	718
notes, cross-staff	326, 330



notine .....	211, 214
notine, chiavi .....	17
notine, formattare le .....	214
<b>null</b> .....	248
<b>NullVoice</b> .....	290
numeri di battuta .....	106
numeri di battuta, cadenze .....	75
numeri di battuta, collisione .....	110
numeri di battuta, con lettere .....	108
numeri di battuta, con ripetizioni .....	108
numeri di battuta, controlli .....	111
numeri di battuta, disposizione a distanza regolare .....	106
numeri di battuta, musica in tempo libero .....	75
numeri di pagina in numeri romani .....	538
numeri di pagina, indicare il primo .....	538
numeri di pagina, numerazione automatica .....	538
numeri di pagina, sopprimere .....	538
numeri di strofa .....	292
<b>numericTimeSignature</b> .....	66
numero del gruppo irregolare, modifiche del .....	49
numero della misura e ripetizioni .....	158
numero di battuta .....	120
numero di battuta, allineamento .....	109
numero di battuta, formato .....	108
numero di ripetizione, modificare .....	158
nuova spaziatura nel corso di un brano .....	566
nuovo rigo .....	190
nuvoletta .....	231
nuvoletta di aiuto .....	231

## O

objects, coloring .....	620
objects, overwriting .....	620
objects, rotating .....	626
objects, visibility of .....	618
<b>octaveCheck</b> .....	10, 786
<b>offset</b> .....	786
oggetti colorati .....	227
oggetti grafici .....	745
oggetti grafici incorporati .....	251
oggetti grafici, disegnare .....	251
oggetti grafici, includere .....	251
oggetti immutabili .....	745
oggetti variabili .....	746
oggetto Scheme .....	747
<b>omit</b> .....	786
on-the-fly .....	488
<b>once</b> .....	786
<b>oneVoice</b> .....	173
open .....	741
open bongo .....	741
open conga .....	741
open high hat .....	741
open string indication .....	336
open timbale .....	741
operazione, inversione .....	14
operazione, inversione modale .....	16
operazione, retrogradazione .....	14
operazione, trasposizione .....	15
operazioni, modali .....	15
oratorio .....	300
orchestral strings .....	335
organo, segni del pedale .....	121

orizzontale, spaziatura .....	564
ornamenti .....	114
ossia .....	200, 205
<b>ottava</b> .....	24, 786
ottava assoluta .....	1
ottava relativa .....	2
ottava relativa e accordi .....	5
ottava relativa e trasposizione .....	5
ottava, controllo .....	10
ottavazione .....	24
<b>Ottoman music</b> .....	469
<b>outer-margin</b> .....	535
output definitions .....	579
output-count .....	746
output-def .....	746
output-suffix .....	746
<b>outside-staff-horizontal-padding</b> .....	563
<b>outside-staff-padding</b> .....	563
<b>outside-staff-priority</b> .....	563
<b>overrideProperty</b> .....	786
overrides, reverting .....	604
<b>overrideTimeSignatureSettings</b> .....	786
overriding for only one moment .....	605
overriding properties within text markup .....	733
overtie-ing text .....	688
overwriting objects .....	620

## P

<b>p</b> .....	124
<b>pad-around</b> .....	252
<b>pad-markup</b> .....	252
<b>pad-to-box</b> .....	252
<b>pad-x</b> .....	252
padding .....	600
padding intorno al testo .....	252
padding text .....	702
padding text horizontally .....	703
page breaking, manual .....	546
<b>page-breaking</b> .....	537
<b>page-breaking-system-system-spacing</b> .....	537
<b>page-count</b> .....	537
<b>page-number-type</b> .....	538
<b>page-spacing-weight</b> .....	538
<b>pageBreak</b> .....	786
<b>pageTurn</b> .....	786
pagina, formato .....	529
pagina, interruzioni .....	569
pagina, orientamento .....	530
<b>palmMute</b> .....	786
<b>palmMuteOn</b> .....	787
Pango .....	257
panning nel MIDI .....	524
<b>paper-height</b> .....	531
<b>paper-width</b> .....	534
parallela, musica .....	187
<b>parallelMusic</b> .....	187, 787
parentesi .....	234
parentesi della volta .....	158
parentesi della volta con testo .....	159
parentesi della volta e legature di valore .....	55
parentesi di raggruppamento delle note .....	234
parentesi graffe, annidamento di .....	194
parentesi orizzontale .....	234
parentesi quadra verticale .....	191

parentesi quadre.....	229	pedale, segni.....	121
parentesi quadre, annidamento di.....	194	pedals, harp.....	334
parentesi uncinate (o angolari).....	167	pedals, piano.....	332
parentesi, fraseggio.....	234	<b>pedalSustainStyle</b> .....	332
parentesi, stile nell'arpeggio attraverso il rigo.....	147	pedice.....	246
<b>parenthesize</b> .....	229, 787	<b>percent</b> .....	162
parlato.....	310	percentuale, ripetizioni.....	162
parlato, testa di nota.....	38	percussion.....	388, 390
parser.....	746	percussion clef.....	389
parser variable.....	746	percussioni, varie.....	741
part songs.....	300	personaggi, nomi dei.....	304
<b>partcombine</b> .....	181, 787	Petrucci.....	432, 433
<b>partcombineApart</b> .....	183	<b>phrasingSlurDashed</b> .....	136
<b>partcombineAutomatic</b> .....	183	<b>phrasingSlurDashPattern</b> .....	136, 787
<b>partcombineChords</b> .....	183	<b>phrasingSlurDotted</b> .....	136
<b>partcombineDown</b> .....	787	<b>phrasingSlurDown</b> .....	136
<b>partcombineForce</b> .....	787	<b>phrasingSlurHalfDashed</b> .....	136
<b>partCombineListener</b> .....	746	<b>phrasingSlurHalfSolid</b> .....	136
<b>partcombineSoloI</b> .....	183	<b>phrasingSlurNeutral</b> .....	136
<b>partcombineSoloII</b> .....	183	<b>phrasingSlurSolid</b> .....	136
<b>partcombineUnisono</b> .....	183	<b>phrasingSlurUp</b> .....	136
<b>partcombineUp</b> .....	787	<b>phrygian</b> .....	22
parte a due.....	181	pi� di pagina.....	477
parte solista.....	181	<b>piano</b> .....	31
<b>partial</b> .....	73, 787	piano e alterazioni.....	31
partitura dentro il blocco markup.....	255	piano music, centering dynamics.....	325
partitura senza i righi vuoti.....	204	piano pedals.....	332
parziale, misura.....	73	piano staves.....	325
paths, drawing.....	714	<i>piano</i> , stile delle alterazioni.....	31
pausa.....	57	<b>piano-cautionary</b> .....	31
pausa d'intero.....	58	<i>piano-cautionary</i> , stile delle alterazioni.....	31
pausa di breve.....	57	pianoforte, rigo per.....	191
pausa di lunga.....	57	<b>PianoStaff</b> .....	325, 328
pausa di maxima.....	57	piatti, vari.....	741
pausa ecclesiastica.....	63	piatto China.....	741
pausa intera per una misura intera.....	61	piatto crash.....	741
pausa invisibile.....	60	piatto ride.....	741
pausa multipla.....	58	piatto splash.....	741
pausa multipla con testo a margine.....	62	<b>Pitch_squash_engraver</b> .....	81
pausa multipla, attaccare fermata.....	62	<b>pitchedTrill</b> .....	149, 787
pausa multipla, attaccare testo.....	62	pitchnames.....	746
pausa multipla, contrazione.....	62	pizzicato.....	741
pausa multipla, espansione.....	62	pizzicato, Bart�k.....	338
pausa multipla, script.....	62	pizzicato, snap.....	338
pausa spaziatrice.....	60	placing horizontal brackets around text.....	712
pausa, collisioni di.....	65	placing parentheses around text.....	713
pausa, inserire le durate.....	57	placing vertical brackets around text.....	708
pausa, specificare la posizione verticale.....	58	<b>pointAndClickOff</b> .....	787
pausa, spostamento automatico.....	177	<b>pointAndClickOn</b> .....	788
pause d'intero.....	61	<b>pointAndClickTypes</b> .....	788
pause multiple.....	61	polifonia su un rigo singolo.....	173
pause multiple e diteggiature.....	65	polifonia, testo cantato condiviso.....	290
pause multiple, posizionamento.....	63	polimetriche, indicazioni.....	76
pause, condensare.....	65	pollice.....	740
pause, divisione.....	79	pollice, indicazione.....	224
PDF metadata.....	489	pollice, segno del (\thumb).....	121
pedal diagrams, harp.....	334	polymetric scores.....	588
pedal high hat.....	741	portamenti indeterminati verso il basso (cadute) e verso l'alto.....	139
pedal indication styles.....	332	portato.....	122, 740
pedal indication, bracket.....	332	posizionamento del testo.....	275
pedal indication, mixed.....	332	posizionamento della parentesi quadra del gruppo irregolare.....	48
pedal indication, text.....	332	posizionamento verticale delle dinamiche.....	127
pedal sustain style.....	332	posizionare pause multiple.....	63
pedal, sostenuto.....	332		
pedal, sustain.....	332		

<b>postscript</b> .....	253
<b>power chords</b> .....	385
<b>powerChords</b> .....	385
<b>pp</b> .....	124
<b>ppp</b> .....	124
<b>pppp</b> .....	124
<b>ppppp</b> .....	124
<b>prall</b> .....	740
<b>pralldown</b> .....	740
<b>prallmordent</b> .....	740
<b>prallprall</b> .....	740
<b>prallup</b> .....	740
<b>predefined string tunings for</b> fretted instruments .....	356
<b>predefinedFretboardsOff</b> .....	380
<b>predefinedFretboardsOn</b> .....	380
<b>prima volta</b> .....	150
<b>principianti, musica</b> .....	39
<b>print-all-headers</b> .....	538
<b>print-first-page-number</b> .....	538
<b>print-page-number</b> .....	538
<b>printing chord names</b> .....	416
<b>printing order</b> .....	620
<b>prob</b> .....	747
<b>properties</b> .....	601
<b>properties, grob</b> .....	603
<b>propertyOverride</b> .....	788
<b>propertyRevert</b> .....	788
<b>propertySet</b> .....	788
<b>propertyTweak</b> .....	788
<b>propertyUnset</b> .....	788
<b>proprietà condivise</b> .....	745
<b>proprietà dell'oggetto</b> .....	747
<b>proprietà di disposizione automatica delle travature</b> per le indicazioni di tempo .....	66
<b>proprietà immutabili</b> .....	745
<b>proprietà variabili</b> .....	746
<b>pull off</b> .....	354
<b>punteggiatura nel testo vocale</b> .....	262
<b>punti coronati</b> .....	740
<b>punto, note</b> .....	46
<b>pure containers, Scheme</b> .....	638
<b>pushToTag</b> .....	788
<b>putting space around text</b> .....	702

## Q

<b>q, chord repetition</b> .....	343
<b>q, ripetizione accordo</b> .....	169
<b>quarter-tones, tab</b> .....	358
<b>quarto di tono</b> .....	6
<b>quotedCueEventTypes</b> .....	213
<b>quotedEventTypes</b> .....	213
<b>quoteDuring</b> .....	211, 214, 788

## R

<b>r</b> .....	57
<b>R</b> .....	61
<b>ragged-bottom</b> .....	531
<b>ragged-last</b> .....	535, 569
<b>ragged-last-bottom</b> .....	531
<b>ragged-right</b> .....	535, 569
<b>raggruppamento dei gruppi irregolari</b> .....	48
<b>raise</b> .....	248
<b>raising text</b> .....	703
<b>Ratisbona, Editio</b> .....	433
<b>referencing contexts</b> .....	581
<b>referencing page labels in text</b> .....	737
<b>referencing page numbers in text</b> .....	734
<b>relativa, ottava</b> .....	2
<b>relative</b> .....	2, 5, 14, 329, 788
<b>relative music and autochange</b> .....	329
<b>relativo</b> .....	2
<b>removals, in chords</b> .....	415
<b>RemoveAllEmptyStaves</b> .....	791
<b>RemoveEmptyStaves</b> .....	792
<b>removeWithTag</b> .....	788
<b>repeatCommands</b> .....	158
<b>repeats in MIDI</b> .....	521
<b>repeatTie</b> .....	55
<b>repetition, using q</b> .....	343
<b>resetRelativeOctave</b> .....	789
<b>respiri</b> .....	137
<b>rest</b> .....	57
<b>rest-event</b> .....	213
<b>restrainOpenStrings</b> .....	343
<b>rests or multi-measure-rests within text by</b> log and dot-count .....	719
<b>rests or multi-measure-rests</b> within text by string .....	719
<b>rests, ancient</b> .....	439
<b>retrogradazione, trasformazione</b> .....	14
<b>retrograde</b> .....	14, 789
<b>reverseturn</b> .....	740
<b>reverting overrides</b> .....	604
<b>revertTimeSignatureSettings</b> .....	789
<b>rfz</b> .....	124
<b>rgb, colore</b> .....	228
<b>rgb-color</b> .....	228
<b>rheel</b> .....	741
<b>RhythmicStaff</b> .....	190
<b>ricopiate, ripetizioni</b> .....	160
<b>ride bell</b> .....	741
<b>ridimensionamento dei righi</b> .....	200
<b>righi annidati</b> .....	194
<b>righi, distanza</b> .....	549
<b>righi, gruppo di</b> .....	191
<b>right aligning text</b> .....	704
<b>right hand fingerings for fretted instruments</b> .....	382
<b>right-align</b> .....	248
<b>right-margin</b> .....	534
<b>rightHandFinger</b> .....	382, 789
<b>rigo Gregoriano per trascrizione</b> .....	190
<b>rigo multiplo</b> .....	191
<b>rigo per batteria</b> .....	190
<b>rigo per coro</b> .....	191
<b>rigo per intavolatura</b> .....	190
<b>rigo per percussioni</b> .....	190
<b>rigo per pianoforte</b> .....	191

rigo ritmico .....	190
rigo temporaneo .....	200, 204
rigo vuoto .....	204
rigo, batteria .....	190
rigo, impostare la dimensione .....	541
rigo, nascondere .....	204
rigo, nuovo .....	190
rigo, percussioni .....	190
rigo, ridimensionamento del .....	200
rigo, simbolo del .....	197
rigo, singolo .....	190
rilegatura .....	535
rinascimentale, musica .....	194
ripetere il testo vocale con finali alternativi .....	285
ripetere le legature di valore .....	55
ripetizione con anacrusi .....	152
ripetizione con finali alternativi .....	150
ripetizione e numero della misura .....	158
ripetizione normale .....	150
ripetizione, breve .....	162
ripetizione, fine .....	158
ripetizione, inizio .....	158
ripetizione, manuale .....	158
ripetizione, tremolo .....	165
ripetizione, uso di q .....	169
ripetizioni alternate .....	160
ripetizioni annidate .....	158
ripetizioni con controlli di battuta .....	152
ripetizioni con legature di valore .....	153
ripetizioni con percentuale .....	162
ripetizioni della misura .....	162
ripetizioni e glissandi .....	158
ripetizioni e informazioni sul tempo .....	158
ripetizioni e legatura di portamento .....	158
ripetizioni e testo vocale .....	281
ripetizioni ricopiate .....	160
ripetizioni, alternativa .....	160
ripetizioni, ambiguità .....	158
ripetizioni, con segno .....	153
ripetizioni, numeri di battuta alternativi .....	156
ripetizioni, numeri di battuta con lettere .....	156
ripetizioni, unfold .....	160
ripristinare le proprietà predefinite delle indicazioni di tempo .....	68
ritmi di accompagnamento per chitarra, mostrare .....	81
ritmi di accompagnamento, mostrare .....	81
ritmi, mostrare la melodia .....	81
ritornelli .....	101
riverbero nel MIDI .....	524
root of chord .....	412
rotating objects .....	626
rotating text .....	704
rounded-box .....	251
rtoe .....	741
rullante .....	741
rullante acustico .....	741
rullante elettrico .....	741

## S

s .....	60
Sacred Harp, testa di nota .....	41
sacredHarpHeads .....	41
sacredHarpHeadsMinor .....	41
saltare le note nel teso vocale .....	286
SATB .....	300
scalable vector graphics, output .....	513
scalare le durate .....	52
scaleDurations .....	53, 76, 789
scaling markup .....	715
scaling text .....	705
scelta della dimensione del tipo di carattere (per gli elementi della notazione) .....	220
Scheme, oggetto .....	747
Scheme, pure containers .....	638
Scheme, unpure containers .....	638
score-markup-spacing .....	533
score-system-spacing .....	533
scoreTitleMarkup .....	485
Scottish highland bagpipe .....	400
script .....	121
script su pausa multipla .....	62
scritta .....	236
scrivere la musica in parallelo .....	187
seconda volta .....	150
segni del pedale .....	121
segni del pedale dell'organo .....	121
segni di chiamata .....	112
segni di tremolo .....	165
segno .....	101, 113, 121, 741
segno del pollice (\thumb) .....	121
segno di bequadro .....	6
segno di chiamata manuale .....	113
segno di chiamata personalizzato .....	113
segno di chiamata, formato .....	113
segno di chiamata, stile .....	113
segno di modifica dell'ottava .....	2
segno di pausa .....	137
segno di spunta .....	138
segno separatore del sistema .....	196
segno sulla stanghetta .....	239
segno, con ripetizioni .....	153
segno, di chiamata, formato .....	113
segno, di chiamata, stile .....	113
self-alignment-X .....	550
Semai form .....	467
semi-bemolle .....	6, 9
semi-diesis .....	6, 9
Semi-flat symbol appearance .....	465
semiaperto .....	741
semicirculus .....	741
sesqui-bemolle .....	9
sesqui-diesis .....	9
set .....	86
set-global-staff-size .....	541
set-octavation .....	24
setting extent of text objects .....	737
setting horizontal text alignment .....	697
setting subscript in standard font size .....	686
setting superscript in standard font size .....	687
settingsFrom .....	789
seventh chords .....	412
sf .....	124

<b>sff</b> .....	124	<b>sp</b> .....	124
<b>sfz</b> .....	124	<b>spacing</b> .....	565
<b>shape</b> .....	789	spacing, vertical .....	549
shaping slurs and ties .....	633	<b>spacingTweaks</b> .....	789
<b>shiftDurations</b> .....	789	<b>Span_stem_engraver</b> .....	330
<b>shiftOff</b> .....	177	spanners, modifying .....	636
<b>shiftOn</b> .....	177	spazi nel testo vocale .....	262
<b>shiftOnn</b> .....	177	spazi, nel testo vocale .....	269
<b>shiftOnnn</b> .....	177	spaziatura del testo .....	279
<b>short-indent</b> .....	208, 536	spaziatura nuova nel corso di un brano .....	566
shortfermata .....	740	spaziatura orizzontale .....	564
<b>show-available-fonts</b> .....	259	spaziatura verticale .....	569
<b>showFirstLength</b> .....	512	spaziatura, visualizzazione della formattazione ...	576
<b>showFirstLength</b> .....	746	spazio bianco .....	476
<b>showKeySignature</b> .....	400	spazio dentro i sistemi .....	549
<b>showLastLength</b> .....	512	spazio tra i righi .....	549
<b>showLastLength</b> .....	746	spezzato, arpeggio .....	144
<b>showStaffSwitch</b> .....	330	spostamento automatico della pausa .....	177
<b>sidestick</b> .....	741	spostare le voci .....	177
<b>signumcongruentiae</b> .....	741	spostare una nota .....	177
sillabe, durata automatica .....	265	<b>spp</b> .....	124
simboli non musicali .....	253	<b>Sprechgesang</b> .....	310
simboli speciali di arpeggio .....	145	spunta, segno di .....	138
simbolo del rigo .....	197	<b>Square neumes ligatures</b> .....	446
simple text strings .....	689	<b>staccatissimo</b> .....	122, 740
simple text strings with tie characters .....	721	<b>staccato</b> .....	122, 740
sincronizzazione degli abbellimenti .....	118	stacking text in a column .....	693
<b>single</b> .....	789	staff change line .....	330
sintassi di LilyPond .....	746	staff changes, automatic .....	328
sintassi di markup .....	243	staff changes, manual .....	326
sistema .....	191	staff switching .....	330
sistema, delimitatori di inizio annidati .....	194	staff symbol, setting of .....	613
sistema, segno separatore .....	196	<b>staff-affinity</b> .....	550
<b>skip</b> .....	60, 789	staff-change line .....	330
<b>skipTypesetting</b> .....	512	<b>staff-staff-spacing</b> .....	550
<b>slashChordSeparator</b> .....	420	<b>Staff.midiInstrument</b> .....	525
slashed digits .....	735	<b>Staff_symbol_engraver</b> .....	204
<b>slashedGrace</b> .....	789	<b>staffgroup-staff-spacing</b> .....	550
slides in tablature notation .....	352	<b>StaffSymbol</b> .....	197
slur-event .....	213	stampare i caratteri riservati .....	244
<b>slurDashed</b> .....	133	stampare i caratteri speciali .....	244
<b>slurDashPattern</b> .....	134, 789	stampo .....	747
<b>slurDotted</b> .....	133	stanghette .....	99
<b>slurDown</b> .....	133	stanghette di chiusura .....	99
<b>slurHalfDashed</b> .....	133	stanghette doppie .....	99
<b>slurHalfSolid</b> .....	133	stanghette invisibili .....	99
<b>slurNeutral</b> .....	133	stanghette manuali .....	99
slurs, modifying .....	632	stanghette predefinite, modifica .....	105
<b>slurSolid</b> .....	133	stanghette, cadenze .....	75
<b>slurUp</b> .....	134	stanghette, definire .....	103
<b>small</b> .....	220, 247	stanghette, musica in tempo libero .....	75
<b>smaller</b> .....	245, 247	stanghette, simboli sulle .....	239
<b>smob</b> .....	747	<b>start-repeat</b> .....	158
snap pizzicato .....	338	<b>startGroup</b> .....	234
snappizzicato .....	741	<b>startStaff</b> .....	197, 200
Sol, chiave di .....	17	<b>startTrillSpan</b> .....	148
Solesmes .....	433	staves, keyboard instruments .....	325
soprano, chiave di .....	17	staves, keyed instruments .....	325
sos. ....	332	staves, piano .....	325
sostenuto pedal. ....	332	<b>Stem</b> .....	330
<b>sostenutoOff</b> .....	332	<b>stem-spacing-correction</b> .....	565
<b>sostenutoOn</b> .....	332	<b>stemDown</b> .....	229
Southern Harmony, testa di nota .....	41	<b>stemLeftBeamCount</b> .....	96
<b>southernHarmonyHeads</b> .....	41	<b>stemNeutral</b> .....	229
<b>southernHarmonyHeadsMinor</b> .....	41	<b>stemRightBeamCount</b> .....	96

stems, cross-staff .....	330
<b>stemUp</b> .....	229
stencil .....	747
stencil, removing .....	619
stereo, bilanciamento nel MIDI .....	524
stile del segno di chiamata .....	113
stile delle alterazioni <i>default</i> .....	29
stile delle alterazioni di	
precauzione <i>modern voice</i> .....	31
stile delle alterazioni <i>dodecafonico</i> .....	32
stile delle alterazioni <i>dodecafonico</i> .....	33
stile delle alterazioni <i>forget</i> .....	34
stile delle alterazioni <i>modern</i> .....	30, 31
stile delle alterazioni <i>modern-cautionary</i> .....	30
stile delle alterazioni <i>modern-voice-cautionary</i> .....	31
stile delle alterazioni <i>neo-modern</i> .....	32
stile delle alterazioni <i>neo-modern-cautionary</i> .....	32
stile delle alterazioni	
<i>neo-modern-voice-cautionary</i> .....	32
stile delle alterazioni <i>no-reset</i> .....	33
stile delle alterazioni per	
insegnamento ( <i>teaching</i> ) .....	33
stile delle alterazioni <i>piano</i> .....	31
stile delle alterazioni <i>piano-cautionary</i> .....	31
stile delle alterazioni <i>voice</i> .....	30
stile delle alterazioni, <i>neo-modern-voice</i> .....	32
stile di alterazione .....	28
stile di alterazione predefinito .....	28
stile dodecafonico neomoderno .....	33
stile moderno delle alterazioni .....	30
stile, legatura di portamento .....	133
stili delle teste di nota .....	38, 681
stili di voce .....	176
<b>stopGroup</b> .....	234
stoppata, nota .....	741
stopped .....	741
<b>stopStaff</b> .....	197, 200, 204
<b>stopTrillSpan</b> .....	148
<b>storePredefinedDiagram</b> .....	373, 789
string numbers .....	336, 339
string vs. fingering numbers .....	339
string, indicating open .....	336
strings, orchestral .....	335
strings, writing for .....	335
<b>stringTuning</b> .....	356, 789
<b>stringTunings</b> .....	356, 369
strofa, numeri di .....	292
strumenti traspositori .....	11
strumenti, nomi .....	525
strumenti, nomi complessi .....	208
strumenti, nomi degli .....	207
strumento traspositore .....	26
<b>styledNoteHeads</b> .....	789
<b>sub</b> .....	246
subbasso, chiave di .....	17
subscript text .....	690
suddivisioni, raggruppamenti .....	92
suggerimenti musicali .....	306
<b>suggestAccidentals</b> .....	440
<b>super</b> .....	246
superscript text .....	690
<b>sus</b> .....	415
sustain pedal .....	332
sustain pedal style .....	332
<b>sustainOff</b> .....	332

<b>sustainOn</b> .....	332
SVG, output .....	513
<b>system-count</b> .....	536
<b>system-separator-markup</b> .....	538
<b>system-system-spacing</b> .....	533
<b>systems-per-page</b> .....	536

## T

tab clef .....	358
tab micro-tones .....	358
tab quarter-tones .....	358
<b>tabChordRepeats</b> .....	790
<b>tabChordRepetition</b> .....	790
tablatura .....	190
tablature .....	339
tablature and harmonic indications .....	346
tablature and slides .....	352
tablature, banjo .....	339, 356, 387
tablature, bass .....	356
tablature, bass guitar .....	356
tablature, cello .....	356
tablature, custom string tunings .....	356
tablature, double bass .....	356
tablature, guitar .....	339, 356
tablature, mandolin .....	356
tablature, predefined string tunings .....	356
tablature, ukulele .....	356
tablature, viola .....	356
tablature, violin .....	356
tablatures, basic .....	341
tablatures, custom .....	356
tablatures, default .....	341
<b>TabStaff</b> .....	190, 341
<b>TabVoice</b> .....	341
tag .....	505
<b>tag</b> .....	790
<b>tagGroup</b> .....	790
tagli addizionali .....	197
tagli addizionali, funzionamento interno .....	197
tagli addizionali, modificare .....	197
tagliata, testa di nota .....	44
tam tam .....	741
tamburello .....	741
tante voci .....	177
<b>taor</b> .....	400
taqasim .....	467
<b>teaching</b> .....	33
<i>teaching</i> , stile delle alterazioni .....	33
<b>teeny</b> .....	220, 247
tempi polimetrici, con travature .....	76
Template Arabic music .....	468
tempo .....	65
<b>tempo</b> .....	70
tempo (all'interno della partitura) .....	120
tempo composto, indicazioni .....	78
tempo polimetrico .....	76
tempo, stile .....	66
<b>temporary</b> .....	790
tenere la musica etichettata .....	505
tenore, chiave di .....	17
tenuto .....	122, 740
terzina, formattazione della .....	49
terzine .....	48

testa di nota tagliata .....	44	testo, dimensione .....	245
testa di nota, Aiken .....	41	testo, incorniciatura .....	251
testa di nota, Christian Harmony .....	41	testo, indicazioni .....	239
testa di nota, forma .....	41	testo, inserimento .....	262
testa di nota, Funk .....	41	testo, padding .....	252
testa di nota, Harmonica Sacra .....	41	testo, posizionamento .....	275
testa di nota, improvvisazione .....	44	testo, tenerlo dentro il margine .....	237
testa di nota, Sacred Harp .....	41	<b>text</b> .....	332
testa di nota, Southern Harmony .....	41	text columns, left-aligned .....	701
testa di nota, Walker .....	41	text columns, right-aligned .....	704
teste di nota .....	220	<b>textLengthOff</b> .....	63, 236
teste di nota a rombo .....	38	<b>textLengthOn</b> .....	63, 236
teste di nota barrate .....	38	<b>textSpannerDown</b> .....	237
teste di nota facili da suonare .....	39	<b>textSpannerNeutral</b> .....	237
teste di nota speciali .....	38	<b>textSpannerUp</b> .....	237
teste di nota, armonico .....	38	Thorough bass .....	424
teste di nota, chitarra .....	38	<b>thumb</b> .....	224
teste di nota, esercizio .....	39	tie-ing text .....	691
teste di nota, notazione semplificata .....	39	<b>tieDashed</b> .....	55
teste di nota, parlato .....	38	<b>tieDashPattern</b> .....	790
teste di nota, stili .....	38, 681	<b>tieDotted</b> .....	55
testo a capo automaticamente .....	250	<b>tieDown</b> .....	55
testo a margine .....	243	<b>tieNeutral</b> .....	55
testo al livello superiore .....	241	ties, modifying .....	632
testo assegnato a una voce .....	173	<b>tieSolid</b> .....	55
testo cantato su abbellimenti .....	294	<b>tieUp</b> .....	55
testo cantato, condiviso tra voci .....	290	timbales .....	741
testo condizionale .....	488	<b>time</b> .....	65, 86, 790
testo esteso su più pagine .....	256	time signature style .....	437
testo formattato su più pagine .....	256	time signature, mensural .....	437
testo fuori dal margine .....	237	time signature, multiple .....	588
testo giustificato .....	250	<b>times</b> .....	790
testo in colonne .....	249	<b>timeSignatureFraction</b> .....	76
testo indipendente .....	241	<b>tiny</b> .....	220, 247
testo multilinea .....	249	tipi di carattere .....	257
testo nella parentesi della volta .....	159	tipi di carattere disponibili, elenco .....	259
testo separato .....	241	tipi di carattere, cambiare .....	245
testo su pausa multipla .....	62	tipi di carattere, famiglie .....	246
testo sulla stanghetta .....	239	tipi di carattere, modificarli per	
testo tra virgolette .....	236	l'intero documento .....	259
testo tra virgolette in modalità markup .....	244	tipi di carattere, trovare quelli disponibili .....	259
testo vocale alternato .....	289	tipo di carattere .....	745
testo vocale diviso .....	289	tipo di carattere Feta .....	660
testo vocale e melodie .....	265	tipo di carattere standard (per gli	
testo vocale e note legate .....	287	elementi della notazione) .....	223
testo vocale, allineamento a una melodia .....	263	tipo di carattere, dimensione .....	245
testo vocale, aumentare lo spazio tra .....	279	tipo di carattere, impostare la dimensione .....	541
testo vocale, citazioni .....	262	tipo di carattere, ridimensionamento .....	220
testo vocale, formattazione .....	262	titoli .....	477
testo vocale, punteggiatura .....	262	<b>tocItem</b> .....	790
testo vocale, ripetizioni .....	281	togliere la musica etichettata .....	505
testo vocale, ripetizioni con finali alternativi .....	285	togliere le citazioni in corpo più piccolo .....	218
testo vocale, saltare le note .....	286	<b>tom</b> .....	741
testo vocale, spazi .....	262	<b>tom alto</b> .....	741
testo vocale, tenerlo dentro il margine .....	237	<b>tom basso</b> .....	741
testo vocale, uso delle variabili .....	273	<b>tom medio</b> .....	741
testo, \skip .....	60	<b>top-margin</b> .....	531
testo, allineamento .....	248	<b>top-markup-spacing</b> .....	533
testo, allineamento orizzontale .....	248	<b>top-system-spacing</b> .....	533
testo, allineamento verticale .....	248	<b>toplevel-bookparts</b> .....	746
testo, altre lingue .....	236	<b>toplevel-scores</b> .....	746
testo, centrare sulla pagina .....	250	tracce MIDI .....	521
testo, comandi di allineamento del .....	251	<b>translate</b> .....	249
testo, con travature .....	86	<b>translate-scaled</b> .....	249
testo, decorazione .....	251	translating text .....	705

transparent, making objects .....	619
<b>transpose</b> .....	5, 11, 14, 790
transposed clefs, visibility of .....	624
<b>transposedCueDuring</b> .....	218, 790
transposing fret diagrams .....	371
<b>transposition</b> .....	26, 211, 790
trascrizione di musica mensurale .....	194
trasformazione retrograda .....	14
trasparenti, note .....	226
trasporre .....	11
traspositori, strumenti .....	11
trasposizione .....	11
trasposizione dell'ottava .....	17
trasposizione delle altezze .....	11
trasposizione delle note .....	11
trasposizione e ottava relativa .....	5
trasposizione MIDI .....	26
trasposizione modale .....	15
trasposizione opzionale dell'ottava .....	17
trasposizione, chiave .....	17
trasposizione, MIDI .....	26
trasposizione, strumento .....	26
trasposizioni modali .....	15
tratti di suddivisione della travatura, direzione ....	91
travatura, direzione dei tratti di suddivisione della .....	91
travatura, estremità in una partitura .....	93
travatura, estremità in voci multiple .....	93
travature a raggiera .....	98
travature angolari .....	84
travature con angolazione .....	84
travature convergenti o divergenti .....	98
travature del tremolo .....	165
travature manuali .....	83, 95
travature manuali, abbellimenti .....	95
travature manuali, scorciatoia per impostare la direzione .....	95
travature, \partcombine con \autoBeamOff .....	85
travature, cadenze .....	75
travature, con melismi .....	84
travature, con tempi polimetrici .....	76
travature, con testo .....	86
travature, interruzioni di linea .....	84
travature, musica in tempo libero .....	75
travature, personalizzazione delle regole .....	83
travature, suddivisione .....	91
tre corde .....	332
<b>treCorde</b> .....	332
<b>tremolo</b> .....	165
tremolo tra due righi .....	166
tremolo, segni .....	165
triads .....	412
<b>triangle</b> .....	253
triangolo .....	741
<b>trill</b> .....	148
trill .....	740
trilli .....	148
trilli con alterazione .....	149
trilli con altezza .....	149
trilli con notina .....	149
trilli con notina e alterazione .....	149
trillo (\trill) .....	121
trovare i tipi di carattere disponibili .....	259
tuning, non-Western .....	463
tunings, banjo .....	388

<b>tuplet</b> .....	48, 76, 790
<b>tupletDown</b> .....	48
<b>tupletNeutral</b> .....	48
<b>TupletNumber</b> .....	49
<b>tupletNumberFormatFunction</b> .....	49
<b>tupletSpan</b> .....	791
<b>tupletSpannerDuration</b> .....	49
<b>tupletUp</b> .....	48
Turkish music .....	469
Turkish note names .....	469
turn .....	740
<b>tweak</b> .....	791
tweak, relation to \override .....	607
tweaking .....	605
tweaking control points .....	607
<b>two-sided</b> .....	535
<b>type</b> .....	594

## U

U.C. ....	332
ukulele .....	360
una corda .....	332
<b>unaCorda</b> .....	332
<b>underline</b> .....	245
underlining text .....	692
undertie-ing text .....	692
<b>undo</b> .....	791
<b>unfold</b> .....	160
unfold, finali alternativi .....	160
unfold, ripetizione .....	160
<b>unfoldRepeats</b> .....	791
<b>unHideNotes</b> .....	226
Unicode .....	509
unione delle parti .....	181
unire le note .....	177
unpure containers, Scheme .....	638
up bow indication .....	336
upbow .....	741
upmordent .....	740
upprall .....	740
UTF-8 .....	509

## V

varbaritono, chiave di .....	17
varcada .....	121, 741
variabile dell'analizzatore sintattico .....	746
variabile globale .....	746
variabile Scheme .....	746
variabili .....	476
variabili, oggetti e proprietà .....	746
variabili, uso delle .....	504
Vaticana, Editio .....	432, 433
<b>VaticanaStaff</b> .....	190
VaticanaStaff .....	442
VaticanaVoice .....	442
vertical spacing .....	549
<b>VerticalAxisGroup</b> .....	550
verticale, spaziatura .....	569
vertically centering text .....	705
verylongfermata .....	740
vibraslap .....	741
violino, chiave di .....	17
virgolette, nel testo vocale .....	269



visibilità dell'indicazione di tempo .....	66
visibility of objects .....	618
visibility of transposed clefs .....	624
voce .....	173
voci divise .....	302
voci multiple .....	177
voci, \partcombine con \autoBeamOff .....	85
voci, citare le .....	211, 214
voci, stili .....	176
<b>voice</b> .....	28, 30
<b>Voice</b> .....	173
voice, following .....	330
<i>voice</i> , stile delle alterazioni .....	30
<b>voiceOne</b> .....	173
<b>void</b> .....	791
volta .....	150
volta della ripetizione, modificare .....	158
volta, parentesi .....	158
volta, prima .....	150
volta, seconda .....	150
volume MIDI .....	516

## W

Walker, testa di nota .....	41
<b>walkerHeads</b> .....	41
<b>walkerHeadsMinor</b> .....	41
<b>whichBar</b> .....	105
whistle .....	741

White mensural ligatures .....	441
wind instruments .....	397
<b>with-color</b> .....	227
<b>withMusicProperty</b> .....	791
woodblock .....	741
<b>wordwrap</b> .....	250
<b>wordwrap-lines</b> .....	256

## X

<b>x-offset</b> .....	550
x11, colore .....	227
<b>x11-color</b> .....	227, 228
<b>xNote</b> .....	791
<b>xNotesOn</b> .....	791