

LilyPond

El gravador de música

Referència de la notació

L'equip de desenvolupadors del LilyPond

Aquest manual ofereix una referència per a tota la notació musical que es pot produir amb el LilyPond versió 2.19.48. S'entén que el lector està familiaritzat amb el material que hi ha a Secció “Manual d'aprenentatge” in *Manual d'aprenentatge*.

Per a més informació sobre la forma en la qual aquest manual es relaciona amb la resta de la documentació, o per llegir aquest manual en altres formats, consulteu Secció “Manuals” in *Informació general*.

Si us falta algun manual, trobareu tota la documentació a <http://www.lilypond.org/>.

Copyright © 1999–2015 pels autors.

La traducció de la següent nota de copyright s'ofereix com a cortesia per a les persones de parla no anglesa, però únicament la nota en anglès té validesa legal.

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

S'atorga permís per copiar, distribuir i/o modificar aquest document sota els termes de la Llicència de Documentació Lliure de GNU, versió 1.1 o qualsevol posterior publicada per la Free Software Foundation; sense cap de les seccions invariants. S'inclou una còpia d'aquesta llicència dins de la secció titulada “Llicència de Documentació Lliure de 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 a la versió del LilyPond 2.19.48

Índex General

1	Notació musical	1
1.1	Altres	1
1.1.1	Espectura de notes	1
	Espectura d'octava absoluta	1
	Espectura d'octava relativa	2
	Alteracions accidentals	5
	Noms de les notes en altes llengües	8
1.1.2	Modificació de diverses notes a l'hora	9
	Comprovació d'octava	9
	Transposició	10
	Inversió	13
	Retrogradació	14
	Transformacions modals	14
1.1.3	Impressió de les altures	17
	Clau	17
	Armadura de la tonalitat	21
	Claudàtors d'octava	24
	Transposició dels instruments	26
	Alteracions accidentals automàtiques	28
	Tessitura	35
1.1.4	Cap de les notes	37
	Caps de nota especials	37
	Caps de notes de Notació Fàcil	39
	Caps de notes amb formes diverses	41
	Improvissació	44
1.2	Rhythms	44
1.2.1	Writing rhythms	45
	Durations	45
	Tuplets	48
	Scaling durations	52
	Ties	53
1.2.2	Writing rests	57
	Rests	57
	Invisible rests	59
	Full measure rests	60
1.2.3	Displaying rhythms	64
	Time signature	65
	Metronome marks	69
	Upbeats	72
	Unmetered music	73
	Polymetric notation	75
	Automatic note splitting	78
	Showing melody rhythms	79
1.2.4	Beams	82
	Automatic beams	82
	Setting automatic beam behavior	84
	Manual beams	93
	Feathered beams	96

1.2.5	Bars	97
	Bar lines	97
	Bar numbers	104
	Bar and bar number checks	109
	Rehearsal marks	110
1.2.6	Special rhythmic concerns	112
	Grace notes	112
	Aligning to cadenzas	117
	Time administration	118
1.3	Expressive marks	119
1.3.1	Expressive marks attached to notes	119
	Articulations and ornamentations	119
	Dynamics	122
	New dynamic marks	128
1.3.2	Expressive marks as curves	130
	Slurs	130
	Phrasing slurs	133
	Breath marks	134
	Falls and doits	136
1.3.3	Expressive marks as lines	137
	Glissando	137
	Arpeggio	142
	Trills	145
1.4	Repeats	147
1.4.1	Long repeats	147
	Normal repeats	148
	Manual repeat marks	155
	Written-out repeats	157
1.4.2	Short repeats	159
	Percent repeats	159
	Tremolo repeats	162
1.5	Simultaneous notes	163
1.5.1	Single voice	164
	Chorded notes	164
	Chord repetition	166
	Simultaneous expressions	168
	Clusters	169
1.5.2	Multiple voices	169
	Single-staff polyphony	169
	Voice styles	173
	Collision resolution	173
	Automatic part combining	178
	Writing music in parallel	183
1.6	Staff notation	186
1.6.1	Displaying staves	186
	Instantiating new staves	186
	Grouping staves	187
	Nested staff groups	191
	Separating systems	193
1.6.2	Modifying single staves	194
	Staff symbol	194
	Ossia staves	197
	Hiding staves	200
1.6.3	Writing parts	202

Instrument names	202
Quoting other voices	206
Formatting cue notes	209
1.7 Editorial annotations	215
1.7.1 Inside the staff	215
Selecting notation font size	215
Fingering instructions	219
Hidden notes	221
Coloring objects	222
Parentheses	223
Stems	224
1.7.2 Outside the staff	225
Balloon help	225
Grid lines	226
Analysis brackets	228
1.8 Text	229
1.8.1 Writing text	230
Text scripts	230
Text spanners	231
Text marks	233
Separate text	236
1.8.2 Formatting text	237
Text markup introduction	237
Selecting font and font size	239
Text alignment	242
Graphic notation inside markup	245
Music notation inside markup	248
Multi-page markup	250
1.8.3 Fonts	251
Fonts explained	251
Single entry fonts	253
Entire document fonts	253
2 Specialist notation	255
2.1 Vocal music	255
2.1.1 Common notation for vocal music	255
References for vocal music	255
Entering lyrics	256
Aligning lyrics to a melody	257
Automatic syllable durations	259
Manual syllable durations	261
Multiple syllables to one note	263
Multiple notes to one syllable	264
Extenders and hyphens	267
2.1.2 Techniques specific to lyrics	267
Working with lyrics and variables	267
Placing lyrics vertically	269
Placing syllables horizontally	273
Lyrics and repeats	275
Divisi lyrics	283
Polyphony with shared lyrics	284
2.1.3 Stanzas	286
Adding stanza numbers	286

Adding dynamics marks to stanzas	286
Adding singers' names to stanzas	287
Stanzas with different rhythms	287
Printing stanzas at the end	290
Printing stanzas at the end in multiple columns	291
2.1.4 Songs	293
References for songs	293
Lead sheets	293
2.1.5 Choral	294
References for choral	294
Score layouts for choral	295
Divided voices	296
2.1.6 Opera and stage musicals	297
References for opera and stage musicals	297
Character names	298
Musical cues	300
Spoken music	303
Dialogue over music	303
2.1.7 Chants psalms and hymns	305
References for chants and psalms	305
Setting a chant	305
Pointing a psalm	312
Partial measures in hymn tunes	315
2.1.8 Ancient vocal music	317
2.2 Keyboard and other multi-staff instruments	317
2.2.1 Common notation for keyboards	318
References for keyboards	318
Changing staff manually	319
Changing staff automatically	320
Staff-change lines	322
Cross-staff stems	323
2.2.2 Piano	325
Piano pedals	325
2.2.3 Accordion	326
Discant symbols	326
2.2.4 Harp	327
References for harps	327
Harp pedals	327
2.3 Unfretted string instruments	328
2.3.1 Common notation for unfretted strings	328
References for unfretted strings	329
Bowing indications	329
Harmonics	330
Snap (Bartók) pizzicato	331
2.4 Fretted string instruments	331
2.4.1 Common notation for fretted strings	332
References for fretted strings	332
String number indications	332
Default tablatures	334
Custom tablatures	348
Fret diagram markups	352
Predefined fret diagrams	362
Automatic fret diagrams	372
Right-hand fingerings	375

2.4.2	Guitar	376
	Indicating position and barring	376
	Indicating harmonics and dampened notes	377
	Indicating power chords	378
2.4.3	Banjo	380
	Banjo tablatures	380
2.4.4	Lute	380
	Lute tablatures	380
2.5	Percussion	381
2.5.1	Common notation for percussion	381
	References for percussion	381
	Basic percussion notation	382
	Drum rolls	383
	Pitched percussion	383
	Percussion staves	383
	Custom percussion staves	386
	Ghost notes	390
2.6	Wind instruments	390
2.6.1	Common notation for wind instruments	391
	References for wind instruments	391
	Fingerings	392
2.6.2	Bagpipes	394
	Bagpipe definitions	394
	Bagpipe example	394
2.6.3	Woodwinds	396
	2.6.3.1 Woodwind diagrams	396
2.7	Chord notation	404
2.7.1	Chord mode	404
	Chord mode overview	404
	Common chords	405
	Extended and altered chords	407
2.7.2	Displaying chords	409
	Printing chord names	409
	Customizing chord names	412
2.7.3	Figured bass	417
	Introduction to figured bass	417
	Entering figured bass	418
	Displaying figured bass	421
2.8	Contemporary music	423
2.8.1	Pitch and harmony in contemporary music	423
	References for pitch and harmony in contemporary music	423
	Microtonal notation	423
	Contemporary key signatures and harmony	423
2.8.2	Contemporary approaches to rhythm	423
	References for contemporary approaches to rhythm	424
	Tuplets in contemporary music	424
	Contemporary time signatures	424
	Extended polymetric notation	424
	Beams in contemporary music	424
	Bar lines in contemporary music	424
2.8.3	Graphical notation	424
2.8.4	Contemporary scoring techniques	424
2.8.5	New instrumental techniques	424
2.8.6	Further reading and scores of interest	424

Books and articles on contemporary musical notation	424
Scores and musical examples	424
2.9 Ancient notation	424
2.9.1 Overview of the supported styles	426
2.9.2 Ancient notation—common features	426
Pre-defined contexts	426
Ligatures	427
Custodes	427
2.9.3 Typesetting mensural music	428
Mensural contexts	428
Mensural clefs	429
Mensural time signatures	430
Mensural note heads	431
Mensural flags	432
Mensural rests	432
Mensural accidentals and key signatures	433
Annotational accidentals (<i>musica ficta</i>)	433
White mensural ligatures	434
2.9.4 Typesetting Gregorian chant	435
Gregorian chant contexts	435
Gregorian clefs	436
Gregorian accidentals and key signatures	437
Divisiones	437
Gregorian articulation signs	438
Augmentum dots (<i>morae</i>)	439
Gregorian square neume ligatures	439
2.9.5 Typesetting Kievan square notation	446
Kievan contexts	446
Kievan clefs	447
Kievan notes	447
Kievan accidentals	448
Kievan bar line	448
Kievan melismata	449
2.9.6 Working with ancient music—scenarios and solutions	450
Incipits	450
Mensurstriche layout	451
Transcribing Gregorian chant	451
Ancient and modern from one source	454
Editorial markings	456
2.10 World music	456
2.10.1 Common notation for non-Western music	456
Extending notation and tuning systems	456
2.10.2 Arabic music	457
References for Arabic music	457
Arabic note names	458
Arabic key signatures	459
Arabic time signatures	460
Arabic music example	461
Further reading for Arabic music	461
2.10.3 Turkish classical music	462
References for Turkish classical music	462
Turkish note names	462

3	General input and output	464
3.1	Input structure	464
3.1.1	Structure of a score	464
3.1.2	Multiple scores in a book	465
3.1.3	Multiple output files from one input file	466
3.1.4	Output file names	467
3.1.5	File structure	468
3.2	Titles and headers	470
3.2.1	Creating titles headers and footers	470
	Titles explained	470
	Default layout of bookpart and score titles	473
	Default layout of headers and footers	476
3.2.2	Custom titles headers and footers	477
	Custom text formatting for titles	477
	Custom layout for titles	478
	Custom layout for headers and footers	481
3.2.3	Creating output file metadata	482
3.2.4	Creating footnotes	483
	Footnotes in music expressions	483
	Footnotes in stand-alone text	488
3.2.5	Reference to page numbers	491
3.2.6	Table of contents	492
3.3	Working with input files	495
3.3.1	Including LilyPond files	495
3.3.2	Different editions from one source	496
	Using variables	496
	Using tags	498
	Using global settings	502
3.3.3	Special characters	502
	Text encoding	502
	Unicode	503
	ASCII aliases	504
3.4	Controlling output	505
3.4.1	Extracting fragments of music	505
3.4.2	Skipping corrected music	505
3.4.3	Alternative output formats	506
3.4.4	Replacing the notation font	506
3.5	Creating MIDI output	507
3.5.1	Supported notation for MIDI	507
3.5.2	Unsupported notation for MIDI	508
3.5.3	The MIDI block	508
3.5.4	Controlling MIDI dynamics	509
	Dynamic marks in MIDI	509
	Setting MIDI volume	510
	Setting MIDI block properties	512
3.5.5	Using MIDI instruments	513
3.5.6	Using repeats with MIDI	514
3.5.7	MIDI channel mapping	515
3.5.8	Context properties for MIDI effects	517
3.5.9	Enhancing MIDI output	518
	The <code>articulate</code> script	518
3.6	Extracting musical information	519
3.6.1	Displaying LilyPond notation	519

3.6.2	Displaying scheme music expressions	519
3.6.3	Saving music events to a file	519
4	Spacing issues	521
4.1	Page layout	521
4.1.1	The <code>\paper</code> block	521
4.1.2	Paper size and automatic scaling	522
	Setting the paper size	522
	Automatic scaling to paper size	523
4.1.3	Fixed vertical spacing <code>\paper</code> variables	523
4.1.4	Flexible vertical spacing <code>\paper</code> variables	524
	Structure of flexible vertical spacing alists	524
	List of flexible vertical spacing <code>\paper</code> variables	525
4.1.5	Horizontal spacing <code>\paper</code> variables	526
	<code>\paper</code> variables for widths and margins	526
	<code>\paper</code> variables for two-sided mode	527
	<code>\paper</code> variables for shifts and indents	528
4.1.6	Other <code>\paper</code> variables	528
	<code>\paper</code> variables for line breaking	529
	<code>\paper</code> variables for page breaking	529
	<code>\paper</code> variables for page numbering	530
	Miscellaneous <code>\paper</code> variables	530
4.2	Score layout	531
4.2.1	The <code>\layout</code> block	531
4.2.2	Setting the staff size	533
4.3	Breaks	535
4.3.1	Line breaking	535
4.3.2	Page breaking	538
	Manual page breaking	538
	Optimal page breaking	539
	Minimal page breaking	539
	One-line page breaking	540
	One-line-auto-height page breaking	540
	Optimal page turning	540
4.4	Vertical spacing	541
4.4.1	Flexible vertical spacing within systems	541
	Within-system spacing properties	542
	Spacing of ungrouped staves	544
	Spacing of grouped staves	546
	Spacing of non-staff lines	547
4.4.2	Explicit staff and system positioning	548
4.4.3	Vertical collision avoidance	555
4.5	Horizontal spacing	556
4.5.1	Horizontal spacing overview	556
4.5.2	New spacing section	558
4.5.3	Changing horizontal spacing	559
4.5.4	Line width	561
4.5.5	Proportional notation	561
4.6	Fitting music onto fewer pages	567
4.6.1	Displaying spacing	568
4.6.2	Changing spacing	569

5	Changing defaults	571
5.1	Interpretation contexts	571
5.1.1	Contexts explained	571
	Output definitions - blueprints for contexts	571
	Score - the master of all contexts	572
	Top-level contexts - staff containers	572
	Intermediate-level contexts - staves	572
	Bottom-level contexts - voices	573
5.1.2	Creating and referencing contexts	573
5.1.3	Keeping contexts alive	576
5.1.4	Modifying context plug-ins	579
5.1.5	Changing context default settings	581
	Changing all contexts of the same type	581
	Changing just one specific context	584
	Order of precedence	585
5.1.6	Defining new contexts	586
5.1.7	Context layout order	588
5.2	Explaining the Internals Reference	590
5.2.1	Navigating the program reference	590
5.2.2	Layout interfaces	591
5.2.3	Determining the grob property	592
5.2.4	Naming conventions	592
5.3	Modifying properties	593
5.3.1	Overview of modifying properties	593
5.3.2	The <code>\set</code> command	593
5.3.3	The <code>\override</code> command	595
5.3.4	The <code>\tweak</code> command	597
5.3.5	<code>\set</code> vs. <code>\override</code>	599
5.3.6	Modifying alists	599
5.4	Useful concepts and properties	601
5.4.1	Input modes	601
5.4.2	Direction and placement	602
	Articulation direction indicators	603
	The direction property	603
5.4.3	Distances and measurements	604
5.4.4	Dimensions	604
5.4.5	Staff symbol properties	605
5.4.6	Spanners	605
	Using the <code>spanner-interface</code>	605
	Using the <code>line-spanner-interface</code>	608
5.4.7	Visibility of objects	610
	Removing the stencil	610
	Making objects transparent	611
	Painting objects white	611
	Using break-visibility	612
	Special considerations	613
5.4.8	Line styles	616
5.4.9	Rotating objects	617
	Rotating layout objects	617
	Rotating markup	617
5.5	Advanced tweaks	618
5.5.1	Aligning objects	618
	Setting <code>X-offset</code> and <code>Y-offset</code> directly	619

Using the <code>side-position-interface</code>	619
Using the <code>self-alignment-interface</code>	619
Using the <code>break-alignable-interface</code>	621
5.5.2 Vertical grouping of grobs.....	623
5.5.3 Modifying stencils	623
5.5.4 Modifying shapes	624
Modifying ties and slurs	624
5.5.5 Modifying broken spanners.....	627
Using <code>\alterBroken</code>	627
5.5.6 Unpure-pure containers	629
5.6 Using music functions.....	631
5.6.1 Substitution function syntax	631
5.6.2 Substitution function examples.....	632

Annex A Notation manual tables 634

A.1 Chord name chart.....	634
A.2 Common chord modifiers.....	635
A.3 Predefined string tunings.....	638
A.4 Predefined fretboard diagrams.....	640
Diagrams for Guitar	640
Diagrams for Ukulele	641
Diagrams for Mandolin	643
A.5 Predefined paper sizes.....	645
A.6 MIDI instruments.....	649
A.7 List of colors	649
A.8 The Feta font	651
Clef glyphs	651
Time Signature glyphs.....	652
Number glyphs	652
Accidental glyphs.....	652
Default Notehead glyphs.....	653
Special Notehead glyphs.....	654
Shape-note Notehead glyphs	654
Rest glyphs.....	658
Flag glyphs.....	659
Dot glyphs	659
Dynamic glyphs	660
Script glyphs	660
Arrowhead glyphs	662
Bracket-tip glyphs	663
Pedal glyphs.....	663
Accordion glyphs	663
Tie glyphs.....	663
Vaticana glyphs	664
Medicaea glyphs.....	665
Hufnagel glyphs	665
Mensural glyphs.....	666
Neomensural glyphs	669
Petrucchi glyphs.....	670
Solesmes glyphs	671
Kievan Notation glyphs.....	671
A.9 Note head styles	672
A.10 Clef styles	673

A.11	Text markup commands	674
A.11.1	Font	674
A.11.2	Align	684
A.11.3	Graphic	699
A.11.4	Music	707
A.11.5	Instrument Specific Markup	713
A.11.6	Accordion Registers	716
A.11.7	Other	721
A.12	Text markup list commands	728
A.13	List of special characters	730
A.14	List of articulations	732
Articulation scripts	732	
Ornament scripts	732	
Fermata scripts	732	
Instrument-specific scripts	733	
Repeat sign scripts	733	
Ancient scripts	733	
A.15	Percussion notes	734
A.16	Technical glossary	736
alist	736	
callback	736	
closure	736	
glyph	737	
grob	737	
immutable	737	
interface	737	
lexer	737	
mutable	738	
output-def	738	
parser	738	
parser variable	738	
prob	738	
smob	739	
stencil	739	
A.17	All context properties	739
A.18	Layout properties	752
A.19	Available music functions	773
A.20	Context modification identifiers	784
A.21	Predefined type predicates	784
R5RS primary predicates	784	
R5RS secondary predicates	784	
Guile predicates	785	
LilyPond scheme predicates	785	
LilyPond exported predicates	785	
A.22	Scheme functions	786
Annex B	Cheat sheet	812
Annex C	GNU Free Documentation License	816
Annex D	Índex d'ordres del LilyPond	823
Annex E	Índex del LilyPond	831

1 Notació musical

Aquest capítol explica com crear notació musical.

1.1 Altures

The image shows a musical score snippet. The top system consists of a piano (p) staff and a bass (b) staff. The piano staff has a treble clef and a common time signature (C). It features a series of chords and single notes, with a dynamic marking of *p* (piano) and a crescendo marking *cresc.* leading to a fortissimo *sf* marking. The bass staff has a bass clef and a common time signature (C). It also features a series of chords and single notes, with a dynamic marking of *p* (piano). The score includes various musical notations such as *dolce e molto legato*, *cresc.*, and *sf*. There are also ledger lines (Led.) with asterisks (*) indicating notes below the staff.

A aquesta secció es discuteix com especificar l'altura de les notes. Aquest procés es compona de tres fases: entrada, modificació i sortida.

1.1.1 Escriptura de notes

A aquesta secció es descriu la manera d'introduir l'altura de les notes. Hi ha dues formes diferents de col·locar les notes a la seva octava corresponent: el mode absolut i el relatiu. A gairebé totes les ocasions, serà més pràctic el mode relatiu.

Escriptura d'octava absoluta

El nom d'una nota s'especifica usant les lletres minúscules de l'a a la g. Les notes els noms dels quals van des de la c fins la b s'imprimeixen a l'octava inferior al Do central.

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

The image shows a musical score snippet. It consists of a single bass staff with a common time signature (C). The staff contains a series of notes: a half note (C4), a quarter note (D4), a quarter note (E4), a quarter note (F4), a quarter note (G4), a quarter note (A4), a quarter note (B4), a quarter note (C5), a quarter note (B4), a quarter note (A4), a quarter note (G4), a quarter note (F4), a quarter note (E4), a quarter note (D4), and a half note (C4).

Es poden especificar d'altres octaves mitjançant una cometa simple quote (') o una coma (,) . Cada ' eleva l'altura en una octava; cada , baixa l'altura una octava.

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



Les marques d'octava comunes es poden escriure una sola vegada sobre una nota de referència si es fa servir `\fixed` abans de la música. A les notes dins de `\fixed` sols els calen les marques d'apòstrof ' o de coma , quan estan per sobre o per sota de l'octava de la nota de referència.

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



L'altura de les notes de l'expressió musical que segueix a `\fixed` no resulta afectada per un `\relative` que l'envolta, que s'estudia a continuació.

Vegeu també

Glossari musical: Secció “Pitch names” in *Glossari musical*.

Fragments de codi: Secció “Altures” in *Fragments de codi*.

Esriptura d'octava relativa

L'entrada d'octava absoluta requereix que s'especifiqui l'octava per a totes i cadascuna de les notes. En contrast amb això, el mode d'entrada d'octava relativa especifica cada octava en relació amb la nota anterior: si es canvia l'octava d'una nota això afectarà a totes les notes següents.

El mode relatiu de notes s'ha d'introduir de forma explícita usant l'ordre `\relative`:

```
\relative altura_inicial expressió_musical
```

En el mode relatiu, se suposa que cada nota es troba el més a prop possible de la nota anterior. Això significa que l'octava d'una nota que està dins de *expresión_musical* es calcula com segueix:

- Si no s'usa cap marca de canvi d'octava en una nota, la seva octava es calcula de forma que l'interval que formi amb la nota anterior sigui menor d'una quinta. Aquest interval es determina sense considerar les alteracions.
- Es pot afegir una marca de canvi d'octava ' o , per elevar o baixar l'altura, respectivament, en una octava més en relació con l'altura calculada sense aquesta marca.
- Es poden usar diverses marques de canvi d'octava. Per exemple, ' ' i , , alteren l'altura en dues octaves.
- L'altura de la primera nota és relativa a *altura_inicial*. *altura_inicial* s'especifica en mode d'octava absoluta. Quines opcions tenen sentit?

c (Do), en qualsevol octava

La identificació del Do central amb c' és quelcom força bàsica, per la qual cosa sol ser fàcil trobar octaves de c. Si la nostra música comença amb gis (un Sol sostingut) per sobre de c'', hauríem d'escriure quelcom com `\relative { gis'' ... }`

una nota que està una o dues octaves de la primera nota

Escriure `\relative { gis'' ... }` fa que sigui fàcil determinar l'altura absoluta de la primera nota de dins.

cap altura d'inici explícita

La forma `\relative { gis'' ... }` serveix com una versió més compacta de l'opció anterior: la primera nota de dins s'escriu ella mateixa en altura absoluta (això resulta ser equivalent a escollir f (Fa) como l'altura de referència).

La documentació sol utilitzar aquesta última opció.

Aquí podem ver el mode relatiu en acció:

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



Les marques de canvi d'octava s'utilitzen per a intervals majors de la quarta:

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



Una sèrie de notes sense cap marca d'octava pot, malgrat tot, abastar intervals molt grans:

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



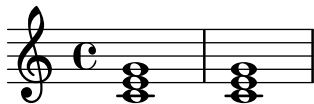
Quan hi ha uns blocs `\relative` niuats dins d'altres, el bloc `\relative` més intern comença amb la seva pròpia nota de referència independentment del `\relative` exterior.

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



`\relative` no té efecte sobre els blocs `\chordmode`.

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



`\relative` no es permet dins des bloques `\chordmode`.

La música que està dins d'un bloc `\transpose` és absoluta, a no ser que s'inclogui una ordre `\relative`.

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



Si l'element anterior és un acord, la primera nota de l'acord s'utilitza per determinar la primera nota del següent acord. Dins dels acords, la següent nota sempre està en relació a l'anterior.

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



Com es va explicar més a dalt, l'octava de les notes es calcula solament a partir dels seus noms, sense tenir en compte cap alteració. Per tant, un Mi doble sostingut després d'un Si s'escriurà més greu. En altres paraules, es considera a la quarta doble augmentada un interval menor que la quinta doble disminuïda, independentment del número de semitons de cada un d'ells.

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



Vegeu també

Glossari musical: Secció “fifth” in *Glossari musical*, Secció “interval” in *Glossari musical*, Secció “Pitch names” in *Glossari musical*.

Referència de la notació: [Comprovació d'octava], pàgina 9.

Fragments de codi: Secció “Pitches” in *Fragments de codi*.

Referència de funcionament intern: Secció “RelativeOctaveMusic” in *Referència de funcionament intern*.

Alteracions accidentals

Nota: Les alteracions accidentals i les armadures són una freqüent causa de confusió per als nous usuaris. Al LilyPond, els noms de les notes específiques les altures; l'armadura i la clau determinen de quin forma es presenten aquestes altures. Una nota sense alteració com c significa ‘Do natural’, i sí que l'afecten l'armadura ni la clau. Per veure més informació, consulteu Secció “Alteracions accidentals i armadures” in *Manual d'aprenentatge*.

S'escriu una nota *sostinguda* afegint **is** al nom de la nota, i un *bemoll* afegint **es**. Com és d'esperar, un *doble sostingut* i un *doble bemoll* s'obté afegint **isis** o **eses**. Aquesta sintaxi deriva dels noms de les notes en holandès. Per utilitzar altres noms per a les alteracions, consulteu [Noms de les notes en altres llengües], pàgina 8.

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



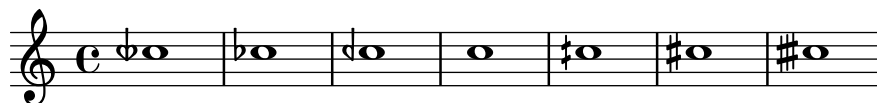
Una nota natural s'introdueix com el nom de la nota, sense més; no cal cap sufix. S'imprimeix un símbol de becaire si cal per cancel·lar l'efecte d'una alteració o armadura anterior.

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



Es poden escriure mig bemolls i els mig sostinguts; a continuació presentem una sèrie de DOs cada com més aguts:

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



Normalment les alteracions accidentals s'imprimeixen automàticament, però també pot imprimir-les manualment. Una alteració recordatòria es pot forçar afegint un signe de admiració ! després de l'altura de la nota. Es pot obtenir una alteració de precaució (o sigui, una alteració entre parèntesi) afegint el signe de interrogació ? després del nom de la nota.

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



Les alteracions sobre notes unides per lligadura sols s'imprimeixen al començament d'un sistema:

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



Fragments de codi seleccionats

Hiding accidentals on tied notes at the start of a new system

This shows how to hide accidentals on tied notes at the start of a new system.

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



Preventing extra naturals from being automatically added

In accordance with traditional typesetting rules, a natural sign is printed before a sharp or flat if a previous double sharp or flat on the same note is canceled. To change this behavior to contemporary practice, set the `extraNatural` property to `f` in the `Staff` context.

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



Vegeu també

Glossari musical: Secció “sharp” in *Glossari musical*, Secció “flat” in *Glossari musical*, Secció “double sharp” in *Glossari musical*, Secció “double flat” in *Glossari musical*, Secció “Pitch names” in *Glossari musical*, Secció “quarter tone” in *Glossari musical*.

Manual d’aprenentatge: Secció “Alteracions accidentals i armadures” in *Manual d’aprenentatge*.

Referència de la notació: [Alteracions accidentals automàtiques], pàgina 28, Secció “Alteracions de anotació (musica ficta)” in *Referència de la notació*, [Noms de les notes en altres llengües], pàgina 8.

Fragments de codi: Secció “Pitches” in *Fragments de codi*.

Referència de funcionament intern: Secció “Accidental_engraver” in *Referència de funcionament intern*, Secció “Accidental” in *Referència de funcionament intern*, Secció “AccidentalCautionary” in *Referència de funcionament intern*, Secció “accidental-interface” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

No hi ha estàndards reconeguts àmpliament per denotar els bemolls de tres quarts, de manera que els símbols del LilyPond no s’ajusten a cap estàndard.

Noms de les notes en altes llengües

Hi ha conjunts predefinitos de noms de notes i les seves alteracions per a algunes llengües a part de l'anglès. La selecció de la llengua dels noms de les notes se sol fer al principi del fitxer: l'exemple següent està escrit utilitzant els noms italians de les notes:

```
\language "italiano"
```

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



Les llengües disponibles i els noms de les notes que defineixen són els següents:

Llengua	Nom de les notes
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 español	do re mi fa sol la sib si
français	do ré/re mi fa sol la sib si
italiano	do re mi fa sol la sib si
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

A més dels noms de les notes, els sufixos de les alteracions poden també variar depenent de la llengua:

Llengua	sostingut	bemoll	doble sostingut	doble bemoll
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 or español	-s	-b	-ss/-x	-bb
français	-d	-b	-dd/-x	-bb
italiano	-d	-b	-dd	-bb
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

A l'holandès, **aes** es contrau com **as**, però les dues formes s'accepten al LilyPond. De forma semblant, s'accepten tant **es** com **ees**. Això s'aplica també a **aeses** / **ases** i a **eeses** / **eses**. A vegades es defineixen sols aquest noms abreujats als fitxers de llengua corresponents.

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



Algunes músiques utilitzen microtons les alteracions dels quals són fraccions d'un sostingut o bemoll 'normals'. La taula següent relaciona els noms de les notes per a les alteracions d'un quart de to en diferents llengües; aquí, els prefixos *semi-* i *sesqui-* respectivament signifiquen 'mig' i 'u i mig'. Les llengües que no apareixen a aquesta taula no aporten encaran noms especials per a les notes.

Llengua	semi-sostingut	semi-bemoll	sesqui-sostingut	sesqui-bemoll
nederlands	-ih	-eh	-isih	-eseh
deutsch	-ih	-eh	-isih	-eseh
english	-qs	-qf	-tqs	-tqf
espanol o español	-cs	-cb	-tcs	-tcb
français	-sd	-sb	-dsd	-bsb
italiano	-sd	-sb	-dsd	-bsb
portugues	-sq	-bq	-stq	-btq

Gairebé totes les llengües presentades aquí estan associades comunament amb la música clàssica occidental, també coneguda com *Període de la pràctica comuna*. Malgrat això, també estan contemplades les altures i els sistemes d'afinació alternatius: vegeu Secció "Notació comuna per a músiques no occidentals" in *Referència de la notació*.

Vegeu també

Glossari musical: Secció "Pitch names" in *Glossari musical*, Secció "Common Practice Period" in *Glossari musical*.

Referència de la notació: Secció "Notació comuna per a músiques no occidentals" in *Referència de la notació*.

Fitxers instal·lats: `scm/define-note-names.scm`.

Fragments de codi: Secció "Pitches" in *Fragments de codi*.

1.1.2 Modificació de diverses notes a l'hora

Aquesta secció tracta de la manera de modificar les altures.

Comprovació d'octava

Al mode relatiu és fàcil oblidar una marca de canvi d'octava. Les comprovacions d'octava fan més fàcil trobar aquests errors, mitjançant la presentació d'un advertiment i corregint l'octava si la nota es troba a una octava diferent de l'esperat.

Per comprovar l'octava d'una nota, especifiqueu l'octava absoluta després del símbol `=`. Aquest exemple genera un missatge d'advertiment (i corregeix l'altura) perquè la segona nota és l'octava absoluta `d''` en lloc de `d'` com indica la correcció d'octava.

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



L'octava de les notes es pot comprovar també amb l'ordre `\octaveCheck` *altura_de_control*. *altura_de_control* s'especifica en mode absolut. Això comprova que l'interval entre la nota anterior i la *altura_de_control* es troba dins d'una quarta (és a dir, el càlcul normal per al mode relatiu). Si aquesta comprovació fracassa, s'imprimeix un missatge d'avertiment. Tot i que la nota prèvia no es modifica, les notes posteriors estan en relació al valor corregit.

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



Compareu els dos compassos següents. La primera i tercera comprovacions de `\octaveCheck` fracassen, però la segona és correcta.

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

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



Vegeu també

Fragments de codi: Secció “Pitches” in *Fragments de codi*.

Referència de funcionament interno: Secció “RelativeOctaveCheck” in *Referència de funcionament intern*.

Transposició

Una expressió musical es pot transportar mitjançant `\transpose`. La sintaxi és

```
\transpose nota_origen nota_destí expressió_musical
```

Això significa que la *expressió_musical* es transporta l'interval que hi ha entre les notes *nota_origen* i *nota_destí*: qualsevol nota amb l'altura de *nota_origen* es canvia per *nota_destí* i qualsevol altra nota es transporta el mateix interval. Les dues notes s'introdueixen en mode absolut.

Nota: La música que està dins d'un bloc `\transpose` és absoluta, a no ser que s'inclogui una introducció `\relative` dins del bloc.

Per exemple, agafem una peça escrita en la tonalitat de Re major. Es pot transposar cap amunt a Mi major; observeu que també l'armadura de la tonalitat es transporta automàticament.

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



Si una partitella escrita en Do (*afinació de concert* normal) es pot tocar amb un clarinet en La (per al qual un La s'escriu com un Do, i que sona una tercera menor per sota del que està escrit), la partitella corresponent es produeix mitjançant:

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



Observeu que especifiquem `\key c \major` de forma explícita. Si no especifiquem cap tonalitat, les notes es transporten però no s'imprimeix l'armadura.

`\transpose` distingeix entre notes enarmòniques: tant `\transpose c cis` com `\transpose c des` transporten un semitò cap amunt. La primera versió imprimeix sostinguts i les notes no es mouen del seu lloc a l'escala, en canvi la segona imprimeix bemolls de la nota següent.

```
music = \relative { c' d e f }
\new Staff {
  \transpose c cis { \music }
  \transpose c des { \music }
}
```



`\transpose` també es pot usar per introduir notes escrites per a un instrument transpositor. Els exemples interiors mostren com escriure altures en Do (o en *afinació de concert*) i gravar-les per a un instrument transpositor, però també és possible el cas contrari si, per exemple, tenim un conjunt de parts instrumentals i volem fer un guió en Do per al director. Per exemple, en introduir música per a trompeta en Si bemoll que comença per un Mi a la partitura (Re de concert), es pot escriure:

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

Per imprimir aquesta música en Fa (per exemple, en arreglar-la per a trompa) podeu envolta la música existent amb un altre `\transpose`:

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

Per veure més informació sobre instruments transpositors, consulteu [Transposició dels instruments], pàgina 26.

Fragments de codi seleccionats

Transposing pitches with minimum accidentals ("Smart" transpose)

This example uses some Scheme code to enforce enharmonic modifications for notes in order to have the minimum number of accidentals. In this case, the following rules apply:

Double accidentals should be removed

B sharp -> C

E sharp -> F

C flat -> B

F flat -> E

In this manner, the most natural enharmonic notes are chosen.

```
#(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 { }
}
```



Vegeu també

Referència de la notació: [Transposició dels instruments], pàgina 26, [Inversió], pàgina 13, [Transformacions modals], pàgina 14, [Escriptura d'octava relativa], pàgina 2, [Retrogradació], pàgina 14.

Fragments de codi: Secció “Pitches” in *Fragments de codi*.

Referència de funcionament intern: Secció “TransposedMusic” in *Referència de funcionament intern*.

Advertisements i problemes coneguts

La conversió relativa no afecta a les seccions `\transpose`, `\chordmode` ni `\relative` dins del seu argument. Per usar el mode relatiu dins de música transportada, es pot col·locar un altre `\relative` dins de `\transpose`.

No s'imprimeixen alteracions accidentals triples quan s'usa `\transpose`. En lloc d'això, s'usarà una nota 'equivalent enarmònicament' (p.ex., Re bemoll en lloc de Mi triple bemoll).

Inversió

Una expressió musical es pot invertir i transportar en una sola operació amb:

\inversion nota_pivot nota_destí expressió_musical

La *expressió_musical* s'inverteix interval a interval al voltant de *nota_pivot*, i després es transport de manera que *nota_pivot* es fa correspondre amb *nota_destí*.

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

```

\music
\inversion d' d' \music
\inversion d' ees' \music
}

```



Nota: Els motius a invertir haurien de venir expressats en mode absolut o convertir-se prèviament a la forma absoluta envoltant-los en un bloc `\relative`.

Vegeu també

Referència de la notació: [Transformacions modals], pàgina 14, [Retrogradació], pàgina 14, [Transposició], pàgina 10.

Retrogradació

Es pot revertir una expressió musical per produir la seva retrogradació:

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

```

\new Staff {
  \music
  \retrograde \music
}

```



Advertiments i problemes coneguts

Les lligadures d'unió manuals dins de `\retrograde` es trencaran i generaran advertiments. Es poden generar algunes lligadures automàticament activant Secció “Divisió automàtica de les notes” in *Referència de la notació*.

Vegeu també

Referència de la notació: [Inversió], pàgina 13, [Transformacions modals], pàgina 14, [Transposició], pàgina 10.

Transformacions modals

En una composició musical que està basada en una escala amb freqüència es transformen els motius de diverses formes. Un motiu es pot *transportar* perquè comenci en diferents llocs de l'escala o pot *invertir-se* al voltant d'una nota pivot de l'escala. També es pot revertir per produir una *retrogradació*, vegeu [Retrogradació], pàgina 14.

Nota: Qualsevol nota que no entri a la escala donada, es deixarà sense transformar.

Transposició modal

Es pot transportar un motiu dins d'una escala donada amb:

```
\modalTranspose nota_origen nota_destí escala motiu
```

Les notes del *motiu* es desplacen dins de la *escala* el nombre de graus de l'escala donats per l'interval entre *nota_destí* y *nota_origen*:

```
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
}
```



Es poden especificar escales ascendents de qualsevol longitud i amb qualssevol intervals:

```
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
}
```



Quan es fa servir amb una escala cromàtica, `\modalTranspose` té un efecte semblant a `\transpose`, però amb la possibilitat d'especificar els noms de les notes que es vulguin usar:

```
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
}
```



Inversió modal

Es pot invertir un motiu dins d'una escala donada al voltant d'una nota pivot donada i transportada al mateix temps en una única operació, amb:

```
\modalInversion nota-pivote nota-destino escala motivo
```

Les notes del *motiu* col·loquen al mateix nombre de graus de distància de l'escala a partir de la *nota_pivot* dins de l'escala, però en la direcció oposada, i el resultat es desplaça després dins de l'escala el nombre de graus de l'escala donats per l'interval entre la *nota_destí* i la *nota_pivot*.

Així doncs, per invertir senzillament al voltant d'una nota de l'escala, utilitzeu el mateix valor per a *nota_pivot* i *nota_destí*:

```
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 invertir al voltant d'una nota pivot entre dues notes de l'escala, inverteix al voltant d'una de les notes i després transporteu en un grau de l'escala. les dues notes especificades es poden interpretar com que fan una forquilla entre elles a la nota pivot:

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

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



L'operació combinada d'inversió i retrogradació produeix la inversió retrògrada:

```
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
}
```



Vegeu també

Referència de la notació: [Inversió], pàgina 13, [Retrogradació], pàgina 14, [Transposició], pàgina 10.

1.1.3 Impressió de les altures

Aquesta secció tracta de com alterar la presentació de l'altura de les notes.

Clau

Sense cap ordre explícita, la clau predeterminada al LilyPond és la clau “treble” (o clau de *Sol*).

```
c'2 c'
```



Tanmateix, es pot canviar la clau usant l'ordre `\clef` i el nom de la clau corresponent. Als exemples següents es mostra la posició del *Do central* en diferents claus.

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



Per veure el repertori complet dels possibles noms per a les claus, consulteu Secció “Estils de clau” in *Referència de la notació*.

Les claus especials, com les utilitzades a la música *antiga*, es descriuen a Secció “Claus de la música mensural” in *Referència de la notació* i a Secció “Claus del cant gregorià” in *Referència de la notació*. La música que requereix claus de tabulatura s'estudia a Secció “Tabulatures pre-determinades” in *Referència de la notació* i a Secció “Tabulatures personalizadas” in *Referència de la notació*.

Per fer servir claus a les notes guia, vegeu les ordres `\cueClef` i `\cueDuringWithClef` a Secció “Formatació de les notes guia” in *Referència de la notació*

En afegir `_8` o `^8` al nom de la clau, la clau es transposa una octava cap avall o cap amunt, respectivament, i `_15` i `^15` la transposa dos octaves. Si cal es poden usar altres nombres enters. L'argument *nome_de_clau* s'ha de envoltar entre cometes si conté caràcters no alfabètics:

```
\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'
```



Es pot obtenir una indicació d'octava opcional envoltant l'argument numèric entre parèntesi o claudàtors rectes:

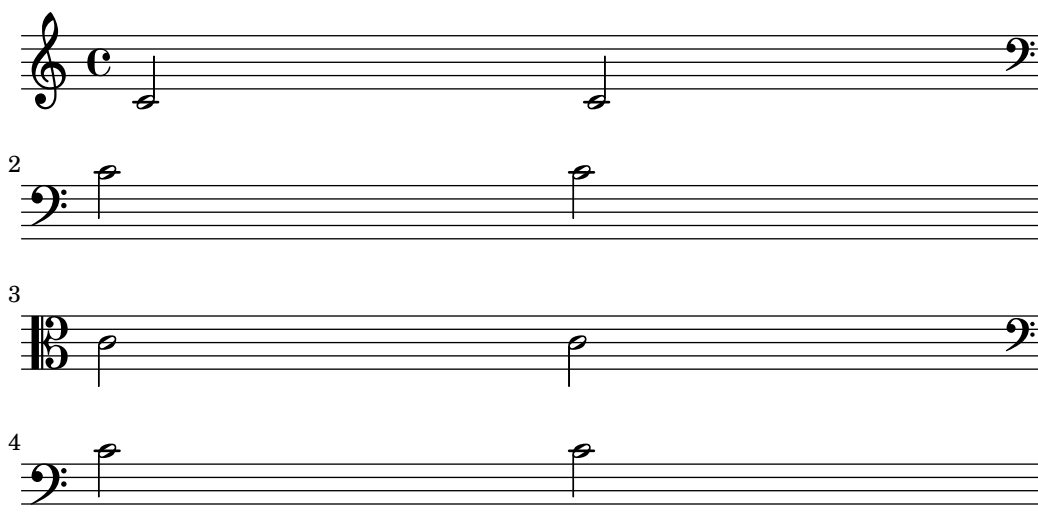
```
\clef "treble_(8)"
c'2 c'
\clef "bass^[15]"
c'2 c'
```



Les altures s'imprimeixen com si l'argument numèric no tingués els parèntesis o claudàtors.

De forma predeterminada, un canvi de clau que es produeix en un salt de línia provoca que el símbol de clau nou s'imprimeixi al final de la línia anterior, com una clau *de cortesia*, així com al principi del següent. Aquesta clau *de cortesia* es pot suprimir.

```
\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
```



De forma predeterminada, una clau que s'ha imprès prèviament no es torna a imprimir si es fa servir un altre cop la mateixa ordre `\clef`, i s'ignora. La instrucció `\set Staff.forceClef = ##t` modifica aquest comportament.

```
\clef treble
```

```

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

```



Quan hi ha un canvi de clau manual, el glif de la clau modificada és més petit del normal. Es pot sobre escriure aquest comportament.

```

\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

```



Fragments de codi seleccionats

Tweaking clef properties

Changing the Clef glyph, its position, or the ottavation does not change the position of subsequent notes on the staff. To get key signatures on their correct staff lines `middleCClefPosition` must also be specified, with positive or negative values moving middle C up or down respectively, relative to the staff's center line.

For example, `\clef "treble_8"` is equivalent to setting the `clefGlyph`, `clefPosition` (the vertical position of the clef itself on the staff), `middleCPosition` and `clefTransposition`. Note that when any of these properties (except `middleCPosition`) are changed a new clef symbol is printed.

The following examples show the possibilities when setting these properties manually. On the first line, the manual changes preserve the standard relative positioning of clefs and notes, whereas on the second line, they do not.

```

{
  % 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

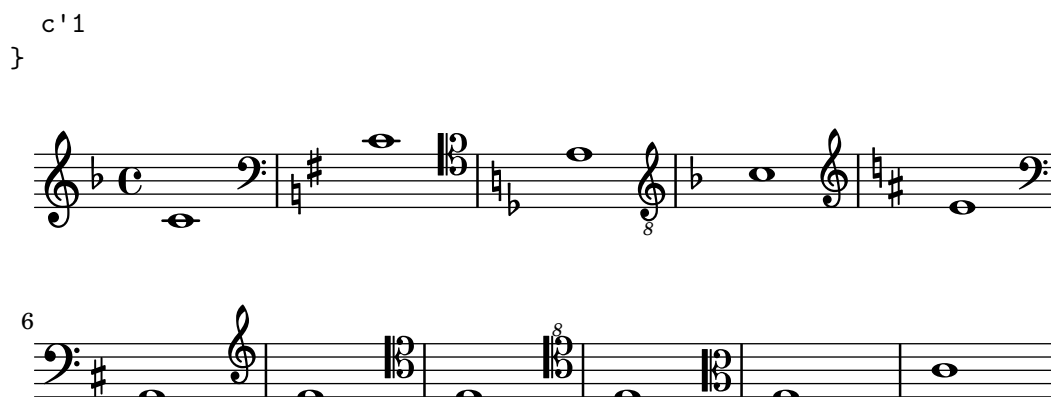
% The following clef changes do not preserve
% the normal relationship between notes, key signatures
% and clefs:

\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

```



Vegeu també

Referència de la notació: Secció “Claus de la música mensural” in *Referència de la notació*, Secció “Claus del cant gregorià” in *Referència de la notació*, Secció “Tabulatures predeterminades” in *Referència de la notació* Secció “Tabulatures personalizadas” in *Referència de la notació*. Secció “Formatació de les notes guia” in *Referència de la notació*.

Fitxers instal·lats: `scm/parser-clef.scm`.

Fragments de codi: Secció “Pitches” in *Fragments de codi*.

Referència de funcionament intern: Secció “Clef_engraver” in *Referència de funcionament intern*, Secció “Clef” in *Referència de funcionament intern*, Secció “ClefModifier” in *Referència de funcionament intern*, Secció “clef-interface” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Els números d’octava adjunts a les claus es tracten com grobs diferents. Així doncs, qualsevol `\override` (sobreescriptura) efectuada a l’objecte `Clef` s’haurà d’aplicar també al grob `ClefModifier` com un `\override` diferent.

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

\clef "treble_8" c'4
```



Armadura de la tonalitat

Nota: Les alteracions accidentals i les armadures són una causa freqüent de confusió per al nous usuaris. Al LilyPond, el noms de les notes són el codi d’entrada en brut; l’armadura i la clau determinen de quina forma es presenta aquest codi en brut. Una nota sense alteració com `c` significa ‘Do natural’, sense que l’afectin l’armadura o la clau. Per veure més informació, consulteu Secció “Altures y armadures” in *Manual d’aprenentatge*.

L'armadura indica la tonalitat en la qual es toca una peça. Està denotada per un conjunt d'alteracions (bemolls o sostinguts) al començament del pentagrama. L'establiment o modificació de l'armadura es fa amb l'ordre `\key`:

```
\key nota modo
```

Aquí, *mode* ha de ser `\major` o `\minor` per obtenir la tonalitat *nota* major o noms estàndard de mode (també coneguts com *modes eclesiàstics*): `\ionian` (jònic), `\dorian` (dòric), `\phrygian` (frigi), `\lydian` (lidi), `\mixolydian` (mixolidi), `\aeolian` (eoli) i `\locrian` (locri).

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



Es poden definir modes addicionals, escrivint una llista amb l'alteració que porta cada nota de l'escala quan el mode comença amb 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
}
```



Les alteracions de l'armadura de la tonalitat es poden imprimir en una octava diferent que en les seves posicions tradicionals, o en més d'una octava, usat les propietats `flat-positions` i `sharp-positions` de `KeySignature`. Les entrades en aquestes propietats especifiquen el rang de posicions de pentagrama en el qual s'imprimiran les alteracions, les alteracions es col·loquen dins de l'octava que acaba en aquesta posició del pentagrama.

```
\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
```



Fragments de codi seleccionats

Preventing natural signs from being printed when the key signature changes

When the key signature changes, natural signs are automatically printed to cancel any accidentals from previous key signatures. This may be prevented by setting to **f** the `printKeyCancellation` property in the **Staff** context.

```
\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
}
```



Non-traditional key signatures

The commonly used `\key` command sets the `keyAlterations` property, in the **Staff** context.

To create non-standard key signatures, set this property directly. The format of this command is a list:

`\set Staff.keyAlterations = #`(((octave . step) . alter) ((octave . step) . alter) ...)` where, for each element in the list, `octave` specifies the octave (0 being the octave from middle C to the B above), `step` specifies the note within the octave (0 means C and 6 means B), and `alter` is `,SHARP`, `,FLAT`, `,DOUBLE-SHARP` etc. (Note the leading comma.)

Alternatively, for each item in the list, using the more concise format `(step . alter)` specifies that the same alteration should hold in all octaves.

For microtonal scales where a “sharp” is not 100 cents, `alter` refers to the alteration as a proportion of a 200-cent whole tone.

Here is an example of a possible key signature for generating a whole-tone scale:

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

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



Vegeu també

Glossari musical: Secció “church mode” in *Glossari musical*, Secció “scordatura” in *Glossari musical*.

Manual de aprenentatge: Secció “Altures i armadures” in *Manual d’aprenentatge*.

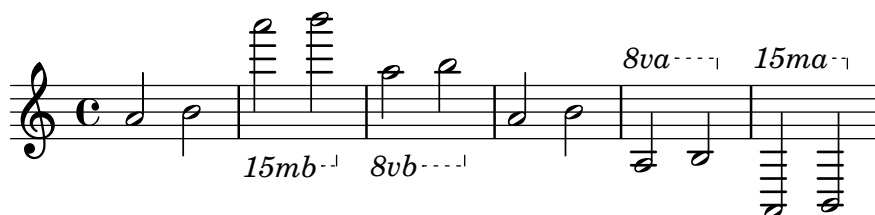
Fragments de codi: Secció “Pitches” in *Fragments de codi*.

Referència de funcionament intern: Secció “KeyChangeEvent” in *Referència de funcionament intern*, Secció “Key_engraver” in *Referència de funcionament intern*, Secció “Key_performer” in *Referència de funcionament intern*, Secció “KeyCancellation” in *Referència de funcionament intern*, Secció “KeySignature” in *Referència de funcionament intern*, Secció “key-signature-interface” in *Referència de funcionament intern*.

Claudàtors d’octava

Els *claudàtors d’Ottava* introdueixen un transport addicional d’una octava per al pentagrama.

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

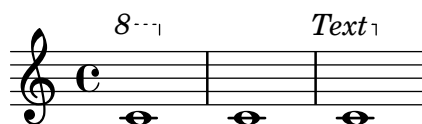


Fragments de codi seleccionats

Ottava text

Internally, `\ottava` sets the properties `ottavation` (for example, to `8va` or `8vb`) and `middleCPosition`. To override the text of the bracket, set `ottavation` after invoking `\ottava`.

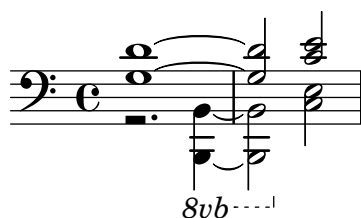
```
{
  \ottava #1
  \set Staff.ottavation = #"8"
  c''1
  \ottava #0
  c'1
  \ottava #1
  \set Staff.ottavation = #"Text"
  c''1
}
```



Adding an ottava marking to a single voice

If you have more than one voice on the staff, setting octavation in one voice will transpose the position of notes in all voices for the duration of the ottava bracket. If the octavation is only intended to apply to one voice, the `middleCPosition` and ottava bracket may be set explicitly. In this snippet, the bass clef usually has `middleCPosition` set to 6, six positions above the center line, so in the 8va portion `middleCPosition` is 7 positions (one octave) higher still.

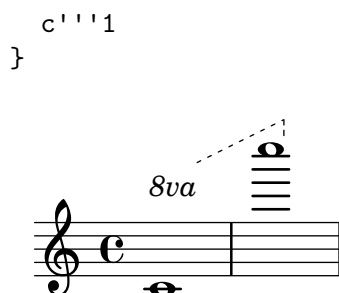
```
{
  \clef bass
  << { <g d'>1~ q2 <c' e'> }
  \\
  {
    r2.
    \set Staff.ottavation = #"8vb"
    \once \override Staff.OttavaBracket.direction = #DOWN
    \set Voice.middleCPosition = #(+ 6 7)
    <b,,, b,,,>4 ~ |
    q2
    \unset Staff.ottavation
    \unset Voice.middleCPosition
    <c e>2
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



Modifying the Ottava spanner slope

It is possible to change the slope of the Ottava spanner.

```
\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
```



Vegeu també

Glossari musical: Secció “octavation” in *Glossari musical*.

Fragments de codi: Secció “Pitches” in *Fragments de codi*.

Referència de funcionament intern: Secció “Ottava_spanner_engraver” in *Referència de funcionament intern*, Secció “OttavaBracket” in *Referència de funcionament intern*, Secció “ottava-bracket-interface” in *Referència de funcionament intern*.

Transposició dels instruments

En gravar partitures on participen instruments transpositors, certes parts es poden gravar en un to diferent del *to de concert*. En aquests casos s’ha d’especificar la tonalitat del *instrumento transpositor*; si no es fa així, la sortida MIDI i les notes guia a altres parts produiran altures incorrectes. Per veure més informació sobre parts citades com a guia, consulteu Secció “Cites d’altres veus” in *Referència de la notació*.

`\transposition pitch`

El to usat per a `\transposition` s’ha de correspondre amb el so real que s’escolta quan l’instrument transpositor interpreta un Do central `c'` escrit al pentagrama. Aquesta nota s’escriu en altura absoluta, per tant un instrument que produeix un so real un to més agut que la música impresa (un instrument en Re) ha d’usar `\transposition d'`. L’ordre `\transposition` s’ha d’usar *solament* si les notes *no* s’escriuran en afinació de concert.

A continuació es poden veure algunes notes per a violí i per a clarinet en Si bemoll, on les parts s’han introduït usant les notes i l’armadura tal i com apareixen a la partitura del director. El que toquen els dos instruments està sonant a l’uníson.

```
\new GrandStaff <<
  \new Staff = "violin" {
    \relative c'' {
      \set Staff.instrumentName = #"Vln"
      \set Staff.midiInstrument = #"violin"
      % no estrictament necessari, però un bon recordatori
      \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
```


Referència de la notació: Secció “Cites d’altres veus” in *Referència de la notació*, [Transposició], pàgina 10.

Fragments de codi: Secció “Pitches” in *Fragments de codi*.

Alteracions accidentals automàtiques

Hi ha moltes convencions diferents sobre la forma de gravar les alteracions. El LilyPond proporciona una funció per especificar quin estil d’alteracions usar. Aquesta funció s’invoca de la manera següent:

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

L’estil d’alteracions s’aplica al **Staff** en curs de forma predeterminada (amb l’excepció dels estils **piano** i **piano-cautionary**, que s’expliquen més endavant). Opcionalment, la funció pot agafar un segon argument que determina en quin àmbit s’ha de canviar l’estil. Per exemple, per usar el mateix estil en tots els pentagrames del **StaffGroup** en curs, useu

```
\accidentalStyle StaffGroup.voice
```

Estan contemplats els següents estils d’alteració. Per donar una mostra de cada u dels estils, utilitzem l’exemple següent:

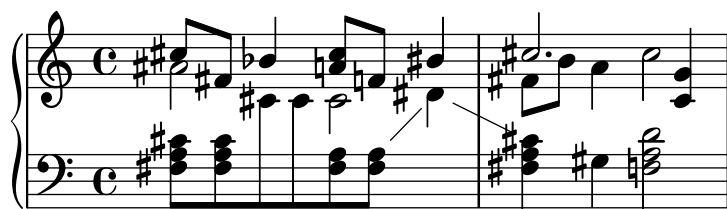
```
musicA = {
  <<
    \relative {
      cis''8 fis, bes4 <a cis>8 f bis4 |
      cis2. <c, g'>4 |
    }
    \\
    \relative {
      ais'2 cis, |
      fis8 b a4 cis2 |
    }
  >>
}

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
    }
  >>
}

```



Observeu que les últimes línies d'aquest exemple es poden substituir per les següents, sempre i quan vulguem usar el mateix estil als dos pentagrames.

```

\new PianoStaff {
  <<
    \context Staff = "up" {
      %% canvieu la línia següent com desitgeu:
      \accidentalStyle Score.default
      \musicA
    }
    \context Staff = "down" {
      \musicB
    }
  >>
}

```

default (predeterminado)

Es el comportament de gravat predeterminat per a composició. Correspon a la pràctica comuna del segle XVIII: les alteracions accidentals es recorden fins al final del compàs en el qual apareixen i sols en la mateixa octava. Així, a l'exemple següent, no s'imprimeixen becaires abans del Si natural al segon compàs ni a l'últim Do:



voice (veu)

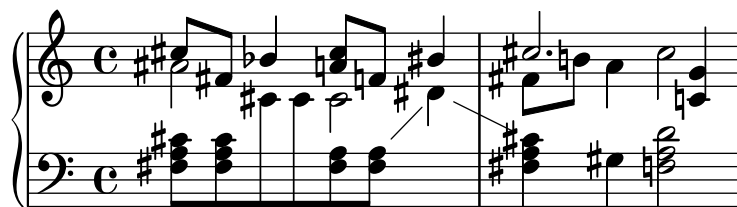
El comportament normal és recordar les alteracions accidentals al nivell de **Staff**. En aquest estil, tanmateix, es graven les alteracions individualment per a cada veu. A part d'això la regla és semblant a **default**.

Com a resultat, les alteracions d'una veu no es cancel·len a les altres veus, el que amb freqüència porta a un resultat no desitjat: a l'exemple següent és difícil determinar si el segon La s'ha de tocar natural o sostingut. Per tant, l'opció **voice** s'ha d'usar sols si les veus es llegiran individualment per músics diferents. Si el pentagrama s'utilitzarà per part d'un sols músic (per exemple un director, o a una partitura de piano), aleshores s'han d'usar en el seu lloc els estils **modern** o **modern-cautionary**.



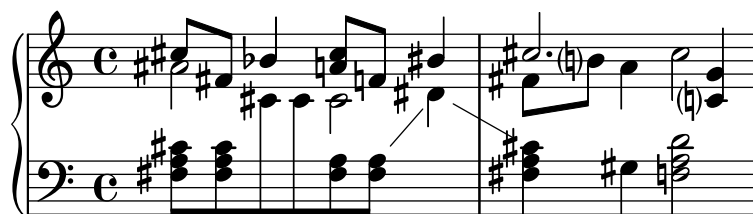
modern

Aquesta regla correspon a la pràctica comuna del segle XX. Omet alguns becaires addicionals, que tradicionalment s'imprimien precedint a un sostingut que segueix a un doble sostingut, o a un bemoll que segueix a un doble bemoll. La regla **modern** imprimeix les mateix alteracions que l'estil **default**, amb dos addicions que serveixen per evitar la ambigüitat: després d'alteracions temporals s'imprimeixen indicacions de cancel·lació també al compàs següent (per a notes a la mateixa octava) i, al mateix compàs, per a notes a octaves diferents. D'aquí els becaires abans del Si natural i del Do al segon compàs del pentagrama superior:



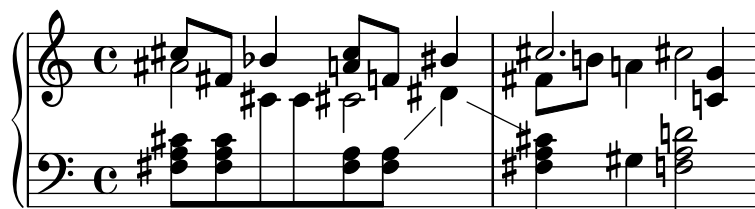
modern-cautionary (modern de precaució)

Aquesta regla és semblant a **modern**, però les alteracions 'afegides' se imprimeixen como alteracions de precaució (entre parèntesis). També es poden imprimir en una mida diferent sobreescrivint la propietat **font-size** de l'objecte **AccidentalCautionary**.



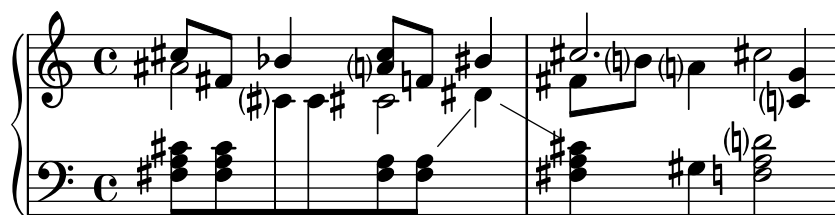
modern-voice (modern, per a veus)

Aquesta regla s'usa perquè puguin llegir les alteracions a diverses veus, tant els músics que toquen una veu com els músics que toquen totes les veus. S'imprimeixen les alteracions per a cada veu, però *es cancel·len* entre veus dins del mateix **Staff**. Per tant, el La a l'últim compàs es cancel·la perquè la cancel·lació anterior estava a una veu diferent, i el Re al pentagrama inferior es cancel·la a causa de l'alteració a una altra veu al compàs previ:



modern-voice-cautionary (modern, veu, de precaució)

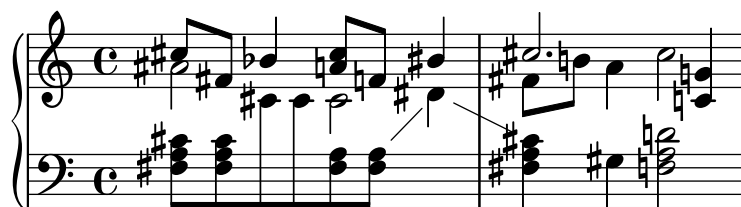
Aquesta regla és la mateixa que `modern-voice`, però amb les alteracions afegides (les que l'estil `voice` no imprimeix) composades com de precaució. Fins i tot malgrat que totes les alteracions impreses per l'estil `default` són impreses amb aquesta regla, algunes d'elles es graven com de precaució.



piano

Aquesta regla reflexa la pràctica del segle XX per a la notació de piano. El seu comportament és molt semblant a l'estil `modern`, però aquí les alteracions també es cancel·len entre diferents pentagrames del mateix grup `GrandStaff` o `PianoStaff`, d'aquí totes les cancel·lacions de les últimes notes.

Aquest estil d'alteració s'aplica de manera predeterminada al grup `GrandStaff` o `PianoStaff` en curs.



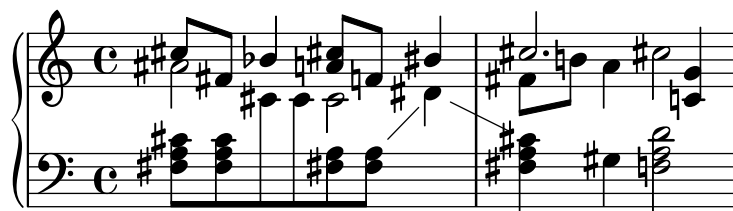
piano-cautionary (piano, de precaució)

Igual que `\accidentalStyle piano` però amb les alteracions afegides composades com de precaució.

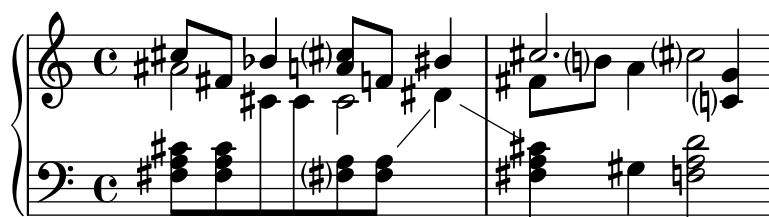


neo-modern

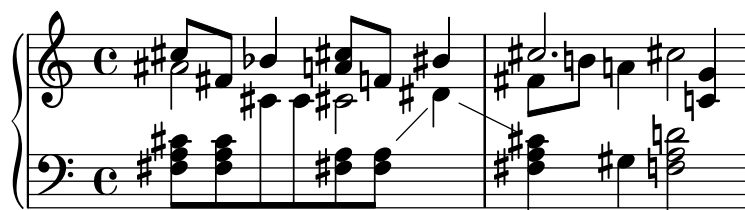
Aquesta regla reproduïx una pràctica comuna a la música contemporània: les alteracions accidentals s'imprimeixen com a `modern`, però es tornen a imprimir si apareix la mateixa nota una altra vegada al mateix compàs (excepte si la nota es repeteix immediatament).

**neo-modern-cautionary**

Aquesta regla és sembla a **neo-modern**, però les alteracions ‘addicionals’ s’imprimeixen com alteracions de precaució (amb parèntesis). També es poden imprimir a una mida diferent sobreescrivint la propietat `font-size` de l’objecte `AccidentalCautionary`.

**neo-modern-voice**

Aquesta regla s’usa per alteracions accidentals sobre diverses veus que s’han de llegir per part de músics que toquen una veu, així com per músics que toquen totes les veus. Les alteracions s’imprimeixen per a cada veu com amb **neo-modern**, però es cancel·len per a altres veus que estan al mateix pentagrama `Staff`.

**neo-modern-voice-cautionary**

Aquesta regla és semblant a **neo-modern-voice**, però les alteracions addicionals s’imprimeixen com alteracions de precaució.

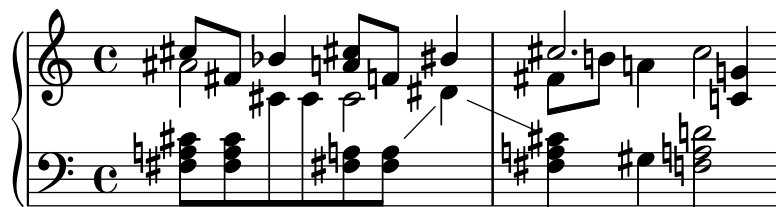
**dodecaphonic**

Aquesta regla reflexa una pràctica introduïda pels compositors de principis del segle XX, en un intent d’abolir la jerarquia entre notes naturals i alterades. Amb aquest estil, *todas* les notes porten alteració, fins i tot becaires.



dodecaphonic-no-repeat

Com amb l'estil d'alteracions dodecafònic *totes* les notes porten una alteració de forma determinada, però les alteracions se suprimeixen quan hi ha notes repetides immediatament al mateix pentagrama.

**dodecaphonic-first**

Semblant a l'estil d'alteracions dodecafònic, *totes* les notes porten una alteració, però sols la primera vegada que es troben al compàs. Les alteracions es recorden sols per a l'octava actual però a través de les veus.

**teaching (ensenyament)**

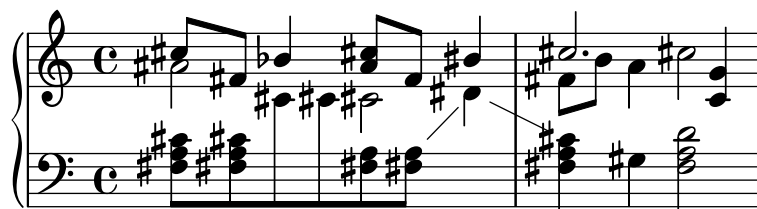
Aquesta regla està pensada per a estudiants, i fa més senzill crear automàticament fulls d'escala amb alteracions de precaució. Les alteracions s'imprimeixen com a l'estil **modern**, però s'afegeixen alteracions de precaució per a totes les notes sostingudes o bemolls especificats per l'armadura, excepte si la nota es repeteix immediatament.

**no-reset (sense restabliment)**

És el mateix que **default** però amb alteracions que duren 'per sempre' i no sols dins del mateix compàs:

**forget (oblit)**

És l'oposat a **sense restabliment**: Les alteracions no es recorden en absolut: d'aquí que totes les alteracions es gravin en relació a l'armadura de la tonalitat, sense que es vegin afectades pel que hi ha abans.



Vegeu també

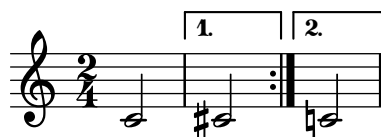
Fragments de codi: Secció “Pitches” in *Fragments de codi*.

Referència de funcionament intern: Secció “Accidental” in *Referència de funcionament intern*, Secció “Accidental_engraver” in *Referència de funcionament intern*, Secció “GrandStaff” in *Referència de funcionament intern*, Secció “PianoStaff” in *Referència de funcionament intern*, Secció “Staff” in *Referència de funcionament intern*, Secció “AccidentalSuggestion” in *Referència de funcionament intern*, Secció “AccidentalPlacement” in *Referència de funcionament intern*, Secció “accidental-suggestion-interface” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

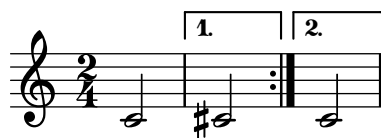
Les notes simultànies no es tenen en compte per a la determinació automàtica de les alteracions accidentals; sols es consideren les notes anteriors i l’armadura de la tonalitat. Pot ser necessari forçar les alteracions accidentals amb `with !` o `?` quan la mateixa nota, amb al mateix nom, aparegui simultàniament amb diferents alteracions, com a `<f! fis!>`.

La cancel·lació de precaució d’alteracions es fa mirant el compàs previ. Malgrat això, al bloc `\alternative` que segueix a una secció de repetició de primera i segona vegada `\repeat volta N`, s’esperaria que la cancel·lació es calculés utilitzant el compàs previ *que s’ha tocat*, no el compàs previ *que s’ha imprès*. A l’exemple següent al Do natural de la casella de segona vegada no li cal becaire:



Es pot usar la següent solució del problema: definir una funció que canviï localment l’estil d’alteracions a `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'
  }
}
```



Tessitura

El terme *ambitus* o àmbit denota el rang de notes que abasta una veu donada a una part musical. També pot denotar el marge de notes que és capaç de tocar un determinat instrument musical. Els àmbits s'imprimeixen a les parts vocals de tal manera que els intèrprets puguin determinar amb facilitat si compleixen amb les seves pròpies possibilitats.

Els àmbits es presenten al començament de la peça junt a la clau inicial. El rang s'especifica gràficament mitjançant dos caps de nota que representen a les notes inferior i superior. Sols s'imprimeixen alteracions si no formen part de l'armadura de la tonalitat.

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

\relative {
  aes' c e2
  cis,1
}
```



Fragments de codi seleccionats

Adding ambitus per voice

Ambitus can be added per voice. In this case, the ambitus must be moved manually to prevent collisions.

```
\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 with multiple voices

Adding the `Ambitus_engraver` to the `Staff` context creates a single ambitus per staff, even in the case of staves with multiple voices.

```
\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
  }
>>
```



Changing the ambitus gap

It is possible to change the default gap between the ambitus noteheads and the line joining them.

```
\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''
}
```



Vegeu també

Glossari musical: Secció “ambitus” in *Glossari musical*.

Fragments de codi: Secció “Pitches” in *Fragments de codi*.

Referència de funcionament intern: Secció “Ambitus-engraver” in *Referència de funcionament intern*, Secció “Voice” in *Referència de funcionament intern*, Secció “Staff” in *Referència de funcionament intern*, Secció “Ambitus” in *Referència de funcionament intern*, Secció “AmbitusAccidental” in *Referència de funcionament intern*, Secció “AmbitusLine” in *Referència de funcionament intern*, Secció “AmbitusNoteHead” in *Referència de funcionament intern*, Secció “ambitus-interface” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

No s'efectua un tractament de les possibles col·lisions en cas de diverses indicacions d'àmbit simultànies a veus diferents.

1.1.4 Cap de les notes

Aquesta secció proposa formes d'alterar els caps de les figures.

Caps de nota especials

Es pot modificar l'aparença del cap de les notes:

```
\relative c'' {
  c4 b
  \override NoteHead.style = #'cross
  c4 b
  \revert NoteHead.style
  a b
```

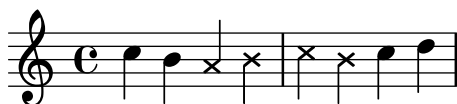
```
\override NoteHead.style = #'harmonic
a b
\revert NoteHead.style
c4 d e f
}
```



Per veure totes els estils de caps de les notes, consulteu Secció “Estils de caps de nota” in *Referència de la notació*.

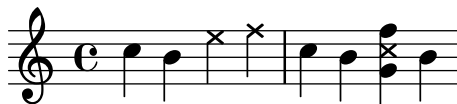
L'estil **cross** (aspes) es fa servir per representar una àmplia varietat d'intencions musicals. Les següent ordres genèriques predefinides modifiquen la forma del cap de les figures tant al context del pentagrama normal com al de tabulatura, i es poden fer servir per representar qualsevol significat musical:

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



La forma de funció musical d'aquesta ordre predefinida es pot usar dins i fora dels acords per generar caps de nota en aspa, tant al context de pentagrama normal com al de tabulatura:

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



Como sinònims de `\xNote`, `\xNotesOn` i `\xNotesOff`, se poden usar `\deadNote`, `\deadNotesOn` i `\deadNotesOff`. El terme *dead note* (nota morta) s'utilitza habitualment per part dels guitarristes.

També hi ha una abreviatura semblant per a les formes en rombe:

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



Instruccions predefinides

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

Vegeu també

Fragments de codi: Secció “Pitches” in *Fragments de codi*.

Referència de la notació: Secció “Estils de caps de nota” in *Referència de la notació*, Secció “Notes en acord” in *Referència de la notació*, Secció “Indicació d’harmònics y notes tapades” in *Referència de la notació*.

Referència de funcionament intern: Secció “note-event” in *Referència de funcionament intern*, Secció “Note_heads_engraver” in *Referència de funcionament intern*, Secció “Ledger_line_engraver” in *Referència de funcionament intern*, Secció “NoteHead” in *Referència de funcionament intern*, Secció “LedgerLineSpanner” in *Referència de funcionament intern*, Secció “note-head-interface” in *Referència de funcionament intern*, Secció “ledger-line-spanner-interface” in *Referència de funcionament intern*.

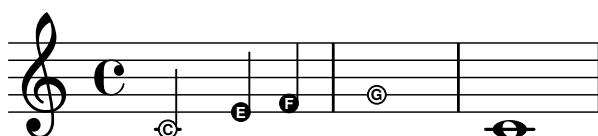
Caps de notes de Notació Fàcil

Les notes amb cap de ‘notació facilitada’ tenen el nom de la nota (en anglès) dins del cap. S’usen a la música per a principiants. Perquè que les lletres siguin llegibles, s’han d’imprimir en una mida gran de tipus de lletra. Per imprimir amb un tipus de lletra més gran, vegeu Secció “Establiment de la mida del pentagrama” in *Referència de la notació*.

```

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

```



Instruccions predefinides

`\easyHeadsOn`, `\easyHeadsOff`.

Fragments de codi seleccionats

Numbers as easy note heads

Easy notation note heads use the `note-names` property of the `NoteHead` object to determine what appears inside the note head. By overriding this property, it is possible to print numbers representing the scale-degree.

A simple engraver can be created to do this for every note head object it sees.

```

#(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)

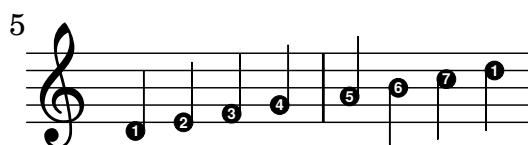
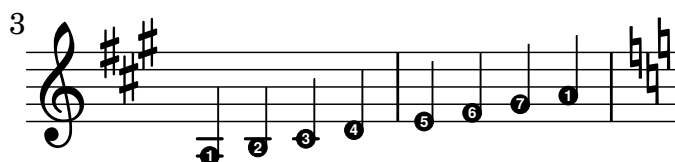
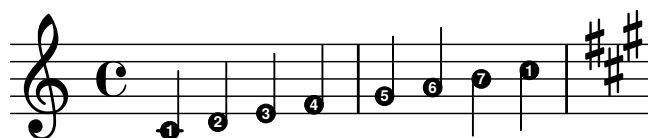
\layout {
  ragged-right = ##t
  \context {
    \Voice
    \consists \Ez_numbers_engraver
  }
}

\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
}

```



Vegeu també

Referència de la notació: Secció “Establiment de la mida del pentagrama” in *Referència de la notació*.

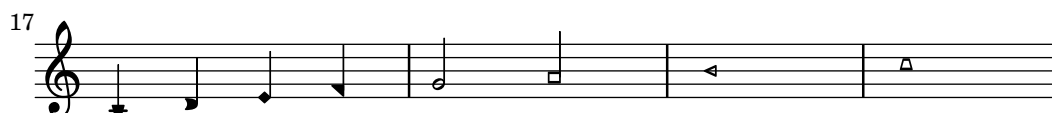
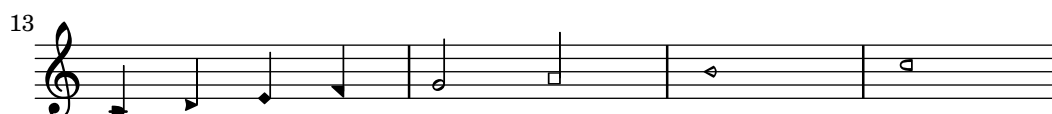
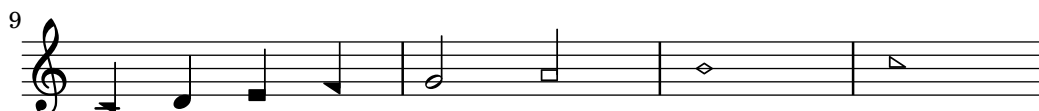
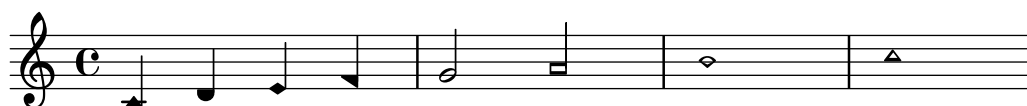
Fragments de codi: Secció “Pitches” in *Fragments de codi*.

Referència de funcionament intern: Secció “note-event” in *Referència de funcionament intern*, Secció “Note_heads_engraver” in *Referència de funcionament intern*, Secció “NoteHead” in *Referència de funcionament intern*, Secció “note-head-interface” in *Referència de funcionament intern*.

Caps de notes amb formes diverses

A la notació de caps amb forma, la forma del cap correspon a la funció harmònica d’una nota dins de l’escala. Aquesta notació es va fer popular als llibres americans de cançons durant el segle XIX. Els caps de nota amb formes es poden produir amb els estils ‘Arpa Sacra’, ‘Southern Harmony’, Funk (Harmonica Sacra), Walker i 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
}
```



Les formes es determinen en funció del grau de l'escala, on la tònica està determinada per l'ordre `\key`. Quan s'escriu en un to menor, la nota de l'escala es pot determinar a partir del relatiu major:

```
\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
}
```

The image displays six staves of musical notation, each representing a different head style for the A minor scale. The staves are numbered 1 through 6 on the left. Each staff begins with a treble clef and a common time signature 'C'. The notes are written on a five-line staff, with accidentals (flats) used for the notes B, D, E, and G. The head styles are: 1. AikenHeads (solid black heads), 2. AikenHeadsMinor (open heads), 3. SacredHarpHeadsMinor (open heads with a small triangle on top), 4. SouthernHarmonyHeadsMinor (open heads with a small triangle on top), 5. FunkHeadsMinor (open heads with a small triangle on top), and 6. WalkerHeadsMinor (open heads with a small triangle on top). The notes are: A (first line), B (first space), C (second line), D (second space), E (third line), F (third space), G (fourth line), and A (fourth space).

Instruccions predefinides

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

Fragments de codi seleccionats

Applying note head styles depending on the step of the scale

The `shapeNoteStyles` property can be used to define various note head styles for each step of the scale (as set by the key signature or the `tonic` property). This property requires a set of symbols, which can be purely arbitrary (geometrical expressions such as `triangle`, `cross`, and `xcircle` are allowed) or based on old American engraving tradition (some latin note names are also allowed).

That said, to imitate old American song books, there are several predefined note head styles available through shortcut commands such as `\aikenHeads` or `\sacredHarpHeads`.

This example shows different ways to obtain shape note heads, and demonstrates the ability to transpose a melody without losing the correspondence between harmonic functions and note head styles.

```

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
  }
}

```



Per veure tots els estils de formes de cap de nota, consulteu Secció “Estils de caps de nota” in *Referència de la notació*,

Vegeu també

Fragments de codi: Secció “Pitches” in *Fragments de codi*.

Referència de la notació: Secció “Estils de caps de nota” in *Referència de la notació*,

Referència de funcionament intern: Secció “note-event” in *Referència de funcionament intern*, Secció “Note_heads_engraver” in *Referència de funcionament intern*, Secció “NoteHead” in *Referència de funcionament intern*, Secció “note-head-interface” in *Referència de funcionament intern*.

Improvisació

La improvisació es denota a vegades mitjançant caps de nota en forma de barra inclinada, on l'executant pot escollir qualsevol nota però amb el ritme especificat. Aquests cap de nota es creen així:

```
\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
}
```



Instruccions predefinides

\improvisationOn, \improvisationOff.

Vegeu també

Fragments de codi: Secció “Pitches” in *Fragments de codi*.

Referència de funcionament intern: Secció “Pitch_squash_engraver” in *Referència de funcionament intern*, Secció “Voice” in *Referència de funcionament intern*, Secció “RhythmicStaff” in *Referència de funcionament intern*.

1.2 Rhythms



Three staves of musical notation for piano, measures 32-34. Measure 32 shows a treble staff with a melodic line and a bass staff with a rhythmic accompaniment. Measure 33 features a piano (*p*) dynamic marking and a crescendo hairpin. Measure 34 continues the melodic and rhythmic patterns with another crescendo marking.

This section discusses rhythms, rests, durations, beaming and bars.

1.2.1 Writing rhythms

Durations

Durations are designated by numbers and dots. Durations are entered as their reciprocal values. For example, a quarter note is entered using a 4 (since it is a $1/4$ note), and a half note is entered using a 2 (since it is a $1/2$ note). For notes longer than a whole you must use the `\longa` (a double breve) and `\breve` commands. Durations as short as 128th notes may be specified. Shorter values are possible, but only as beamed notes.

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

A single staff of musical notation in 8/1 time, showing the same sequence of durations as the code block above. The notes are: a double-clef C (c''), a longa (c), a breve ($c\breve$), a quarter note (c_1), an eighth note (c_2), and a series of beamed sixteenth, thirty-second, sixty-fourth, and one-hundred-twenty-eighth notes ($c_4 c_8 c_{16} c_{32} c_{64} c_{128} c_{128}$).

Here are the same durations with automatic beaming turned off.

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



A note with the duration of a quadruple breve may be entered with `\maxima`, but this is supported only within ancient music notation. For details, see Secció 2.9 [Ancient notation], pàgina 424.

If the duration is omitted, it is set to the previously entered duration. The default for the first note is a quarter note.

```
\relative { a' a a2 a a4 a a1 a }
```



Durations occurring on their own within a music sequence will take their pitches from the preceding note or chord.

```
\relative {
  \time 8/1
  c'' \longa \breve 1 2
  4 8 16 32 64 128 128
}
```



To obtain dotted note lengths, place a dot (.) after the duration. Double-dotted notes are specified by appending two dots, and so on.

```
\relative { a'4 b c4. b8 a4. b4.. c8. }
```



Some durations cannot be represented with just binary durations and dots; they can be represented only by tying two or more notes together. For details, see [Ties], pàgina 53.

For ways of specifying durations for the syllables of lyrics and ways of aligning lyrics to notes, see Secció 2.1 [Vocal music], pàgina 255.

Optionally, notes can be spaced strictly proportionately to their duration. For details of this and other settings which control proportional notation, see Secció 4.5.5 [Proportional notation], pàgina 561.

Dots are normally moved up to avoid staff lines, except in polyphonic situations. Dots may be manually placed above or below the staff; see Secció 5.4.2 [Direction and placement], pàgina 602.

Instruccions predefinides

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


Fragments de codi seleccionats

Alternative breve notes

Breve notes are also available with two vertical lines on each side of the notehead instead of one line and in baroque style.

```
\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
}
```



Changing the number of augmentation dots per note

The number of augmentation dots on a single note can be changed independently of the dots placed after the note.

```
\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 |
}
```



Vegeu també

Music Glossary: Secció “breve” in *Glossari musical*, Secció “longa” in *Glossari musical*, Secció “maxima” in *Glossari musical*, Secció “note value” in *Glossari musical*, Secció “Duration names notes and rests” in *Glossari musical*.

Notation Reference: [Automatic beams], pàgina 82, [Ties], pàgina 53, [Stems], pàgina 224, Secció 1.2.1 [Writing rhythms], pàgina 45, Secció 1.2.2 [Writing rests], pàgina 57, Secció 2.1 [Vocal music], pàgina 255, Secció 2.9 [Ancient notation], pàgina 424, Secció 4.5.5 [Proportional notation], pàgina 561.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “Dots” in *Referència de funcionament intern*, Secció “DotColumn” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

There is no fundamental limit to rest durations (both in terms of longest and shortest), but the number of glyphs is limited: rests from 128th to maxima ($8\times$ whole) may be printed.

Tuplets

Tuplets are made from a music expression with the `\tuplet` command, multiplying the speed of the music expression by a fraction:

```
\tuplet fraction { music }
```

The fraction's numerator will be printed over or under the notes, optionally with a bracket. The most common tuplets are triplets (3 notes sound within the duration normally allowed for 2).

```
\relative {
  a'2 \tuplet 3/2 { b4 4 4 }
  c4 c \tuplet 3/2 { b4 a g }
}
```



When entering long passages of tuplets, having to write a separate `\tuplet` command for each group is inconvenient. It is possible to specify the duration of one tuplet group directly before the music in order to have the tuplets grouped automatically:

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



Tuplet brackets may be manually placed above or below the staff:

```
\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 }
}
```



Tuplets may be nested:

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



Modifying nested tuplets which begin at the same musical moment must be done with `\tweak`.

To modify the duration of notes without printing a tuplet bracket, see [Scaling durations], pàgina 52.

Instruccions predefinides

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

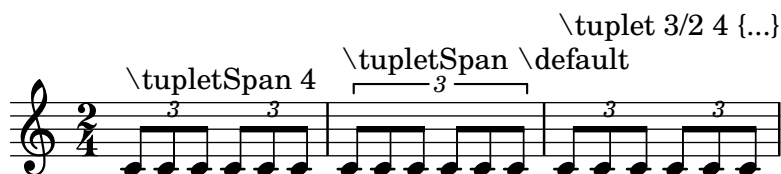
Fragments de codi seleccionats

Entering several tuplets using only one `\tuplet` command

The property `tupletSpannerDuration` sets how long each of the tuplets contained within the brackets after `\tuplet` should last. Many consecutive tuplets can then be placed within a single `\tuplet` expression, thus saving typing.

There are several ways to set `tupletSpannerDuration`. The command `\tupletSpan` sets it to a given duration, and clears it when instead of a duration `\default` is specified. Another way is to use an optional argument with `\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 }
}
```



Changing the tuplet number

By default, only the numerator of the tuplet number is printed over the tuplet bracket, i.e., the numerator of the argument to the `\tuplet` command.

Alternatively, num:den of the tuplet number may be printed, or the tuplet number may be suppressed altogether.

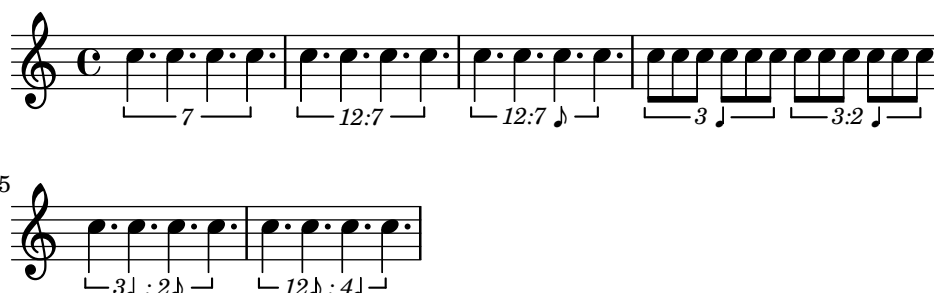
```
\relative c'' {
  \tuplet 3/2 { c8 c c }
  \tuplet 3/2 { c8 c c }
  \override TupletNumber.text = #tuplet-number::calc-fraction-text
  \tuplet 3/2 { c8 c c }
  \omit TupletNumber
  \tuplet 3/2 { c8 c c }
}
```



Non-default tuplet numbers

LilyPond also provides formatting functions to print tuplet numbers different than the actual fraction, as well as to append a note value to the tuplet number or tuplet fraction.

```
\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. }
}
```

*Controlling tuplet bracket visibility*

The default behavior of tuplet-bracket visibility is to print a bracket unless there is a beam of the same length as the tuplet. To control the visibility of tuplet brackets, set the property 'bracket-visibility to either **#t** (always print a bracket), **#f** (never print a bracket) or **#'if-no-beam** (only print a bracket if there is no beam).

```
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" >>
}
}

```



Permitting line breaks within beamed triplets

This artificial example shows how both manual and automatic line breaks may be permitted to within a beamed triplet. Note that such off-beat triplets have to be beamed manually.

```

\layout {
  \context {
    \Voice
    % Permit line breaks within triplets
    \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
}

```



Vegeu també

Music Glossary: Secció “triplet” in *Glossari musical*, Secció “tuplet” in *Glossari musical*, Secció “polymetric” in *Glossari musical*.

Learning Manual: Secció “Tweaking methods” in *Manual d’aprenentatge*.

Notation Reference: Secció 5.4.2 [Direction and placement], pàgina 602, [Time administration], pàgina 118, [Scaling durations], pàgina 52, Secció 5.3.4 [The tweak command], pàgina 597, [Polymetric notation], pàgina 75.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “TupletBracket” in *Referència de funcionament intern*, Secció “TupletNumber” in *Referència de funcionament intern*, Secció “TimeScaledMusic” in *Referència de funcionament intern*.

Scaling durations

The duration of single notes, rests or chords may be multiplied by a fraction N/M by appending $*N/M$ (or $*N$ if M is 1) to the duration. This will not affect the appearance of the notes or rests produced, but the altered duration will be used in calculating the position within the measure and setting the duration in the MIDI output. Multiplying factors may be combined like $*L*M/N$. Factors are part of the duration: if a duration is not specified for subsequent notes, the default duration taken from the preceding note will include any scaling factor.

In the following example, the first three notes take up exactly two beats, but no triplet bracket is printed.

```

\relative {
  \time 2/4
  % Alter durations to triplets
  a'4*2/3 gis a
  % Normal durations
  a4 a
  % Double the duration of chord
  <a d>4*2
  % Duration of quarter, appears like sixteenth
  b16*4 c4
}

```



The duration of spacer rests may also be modified by a multiplier. This is useful for skipping many measures, e.g., `s1*23`.

Longer stretches of music may be compressed by a fraction in the same way, as if every note, chord or rest had the fraction as a multiplier. This leaves the appearance of the music unchanged but the internal duration of the notes will be multiplied by the fraction *num/den*. Here is an example showing how music can be compressed and expanded:

```
\relative {
  \time 2/4
  % Normal durations
  <c' a>4 c8 a
  % Scale music by *2/3
  \scaleDurations 2/3 {
    <c a f>4. c8 a f
  }
  % Scale music by *2
  \scaleDurations 2/1 {
    <c' a>4 c8 b
  }
}
```



One application of this command is in polymeric notation, see [Polymetric notation], pàgina 75.

Vegeu també

Notation Reference: [Tuplets], pàgina 48, [Invisible rests], pàgina 59, [Polymetric notation], pàgina 75.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Advertiments i problemes coneguts

The calculation of the position within a measure must take into account all the scaling factors applied to the notes within that measure and any fractional carry-out from earlier measures. This calculation is carried out using rational numbers. If an intermediate numerator or denominator in that calculation exceeds 2^{30} the execution and typesetting will stop at that point without indicating an error.

Ties

A tie connects two adjacent note heads of the same pitch. The tie in effect extends the duration of a note.

Nota: Ties should not be confused with *slurs*, which indicate articulation, or *phrasing slurs*, which indicate musical phrasing. A tie is just a way of extending a note duration, similar to the augmentation dot.

A tie is entered by appending a tilde symbol (~) to the first of each pair of notes being tied. This indicates that the note should be tied to the following note, which must be at the same pitch.

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



Ties can make use of the ‘last explicit pitch’ interpretation of isolated durations:

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



Ties are used either when the note crosses a bar line, or when dots cannot be used to denote the rhythm. Ties should also be used when note values cross larger subdivisions of the measure:

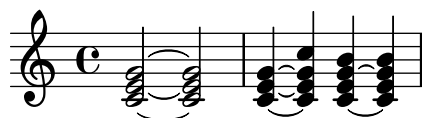
```
\relative {
  r8 c'~ 2 r4 |
  r8~"not" c2~ 8 r4
}
```



If you need to tie many notes across bar lines, it may be easier to use automatic note splitting, see [Automatic note splitting], pàgina 78. This mechanism automatically splits long notes, and ties them across bar lines.

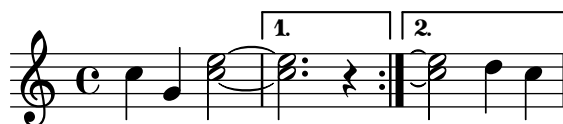
When a tie is applied to a chord, all note heads whose pitches match are connected. When no note heads match, no ties will be created. Chords may be partially tied by placing the ties inside the chord.

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



When a second alternative of a repeat starts with a tied note, you have to specify the repeated tie as follows:

```
\relative {
  \repeat volta 2 { c'' g <c e>2~ }
  \alternative {
    % First alternative: following note is tied normally
    { <c e>2. r4 }
    % Second alternative: following note has a repeated tie
    { <c e>2\repeatTie d4 c }
  }
}
```

L.v. ties (*laissez vibrer*) indicate that notes must not be damped at the end. It is used in notation for piano, harp and other string and percussion instruments. They can be entered as follows:

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



Ties may be made to curve up or down manually; see Secció 5.4.2 [Direction and placement], pàgina 602.

Ties may be made dashed, dotted, or a combination of solid and dashed.

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



Custom dash patterns can be specified:

```
\relative c' {
  \tieDashPattern #0.3 #0.75
  c2~ 2
  \tieDashPattern #0.7 #1.5
  c2~ 2
  \tieSolid
  c2~ 2
}
```



Dash pattern definitions for ties have the same structure as dash pattern definitions for slurs. For more information about complex dash patterns, see [Slurs], pàgina 130.

Override *whiteout* and *layer* layout properties of objects that should cause a gap in ties.

```
\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
}

```



Instruccions predefinides

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

Fragments de codi seleccionats

Using ties with arpeggios

Ties are sometimes used to write out arpeggios. In this case, two tied notes need not be consecutive. This can be achieved by setting the `tieWaitForNote` property to `##t`. The same feature is also useful, for example, to tie a tremolo to a chord, but in principle, it can also be used for ordinary consecutive notes.

```

\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
}

```



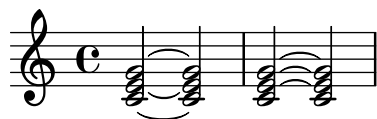
Engraving ties manually

Ties may be engraved manually by changing the `tie-configuration` property of the `TieColumn` object. The first number indicates the distance from the center of the staff in half staff-spaces, and the second number indicates the direction (1 = up, -1 = down).

```

\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>
}

```



Vegeu també

Music Glossary: Secció “tie” in *Glossari musical*, Secció “laissez vibrer” in *Glossari musical*.

Notation Reference: [Slurs], pàgina 130, [Automatic note splitting], pàgina 78.

Snippets: Secció “Expressive marks” in *Fragments de codi*, Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “LaissezVibrerTie” in *Referència de funcionament intern*, Secció “LaissezVibrerTieColumn” in *Referència de funcionament intern*, Secció “TieColumn” in *Referència de funcionament intern*, Secció “Tie” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Switching staves when a tie is active will not produce a slanted tie.

Changing clefs or ottavations during a tie is not really well-defined. In these cases, a slur may be preferable.

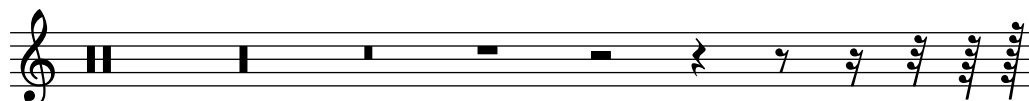
1.2.2 Writing rests

Rests are entered as part of the music in music expressions.

Rests

Rests are entered like notes with the note name `r`. Durations longer than a whole rest use the following predefined commands:

```
\new Staff {
  % These two lines are just to prettify this example
  \time 16/1
  \omit Staff.TimeSignature
  % Print a maxima rest, equal to four breves
  r\maxima
  % Print a longa rest, equal to two breves
  r\longa
  % Print a breve rest
  r\breve
  r1 r2 r4 r8 r16 r32 r64 r128
}
```



Whole measure rests, centered in the middle of the measure, must be entered as multi-measure rests. They can be used for a single measure as well as many measures and are discussed in [Full measure rests], pàgina 60.

To explicitly specify a rest’s vertical position, write a note followed by `\rest`. A rest of the duration of the note will be placed at the staff position where the note would appear. This allows for precise manual formatting of polyphonic music, since the automatic rest collision formatter will not move these rests.

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



Fragments de codi seleccionats

Rest styles

Rests may be used in various styles.

```
\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
}
```

The image displays four musical staves, each illustrating a different rest style. The first staff, labeled 'mensural', shows a series of horizontal lines of varying lengths representing rests. The second staff, labeled 'neomensural', shows rests as vertical lines of varying heights. The third staff, labeled 'classical', shows rests as horizontal lines of varying lengths, with some rests having a small 'x' mark. The fourth staff, labeled 'z-style', shows rests as horizontal lines of varying lengths, with some rests having a small 'x' mark.



Vegeu també

Music Glossary: Secció “breve” in *Glossari musical*, Secció “longa” in *Glossari musical*, Secció “maxima” in *Glossari musical*.

Notation Reference: [Full measure rests], pàgina 60.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “Rest” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

There is no fundamental limit to rest durations (both in terms of longest and shortest), but the number of glyphs is limited: there are rests from 128th to maxima ($8 \times$ whole).

Invisible rests

An invisible rest (also called a ‘spacer rest’) can be entered like a note with the note name `s`:

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



Spacer rests are available only in note mode and chord mode. In other situations, for example, when entering lyrics, the command `\skip` is used to skip a musical moment. `\skip` requires an explicit duration, but this is ignored if the lyrics derive their durations from the notes in an associated melody through `\addlyrics` or `\lyricsto`.

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



Because `\skip` is a command, it does not affect the default durations of following notes, unlike `s`.

```
<<
{
  \repeat unfold 8 { a'4 }
}
```

```
{
  a'4 \skip 2 a' |
  s2 a'
}
```

>>



A spacer rest implicitly causes **Staff** and **Voice** contexts to be created if none exist, just like notes and rests do:

```
{ s1 s s }
```



`\skip` simply skips musical time; it creates no output of any kind.

```
% This is valid input, but does nothing
{ \skip 1 \skip1 \skip 1 }
```

Vegeu també

Learning Manual: Secció “Visibility and color of objects” in *Manual d’aprenentatge*.

Notation Reference: [Hidden notes], pàgina 221, Secció 5.4.7 [Visibility of objects], pàgina 610.

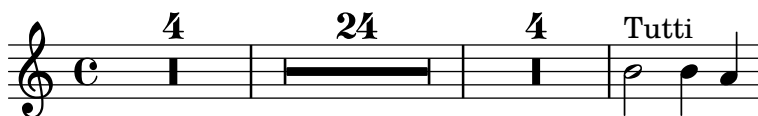
Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “SkipMusic” in *Referència de funcionament intern*.

Full measure rests

Rests for one or more full measures are entered like notes with the note name uppercase R:

```
% Rest measures contracted to single measure
\compressMMRests {
  R1*4
  R1*24
  R1*4
  b'2~"Tutti" b'4 a'4
}
```



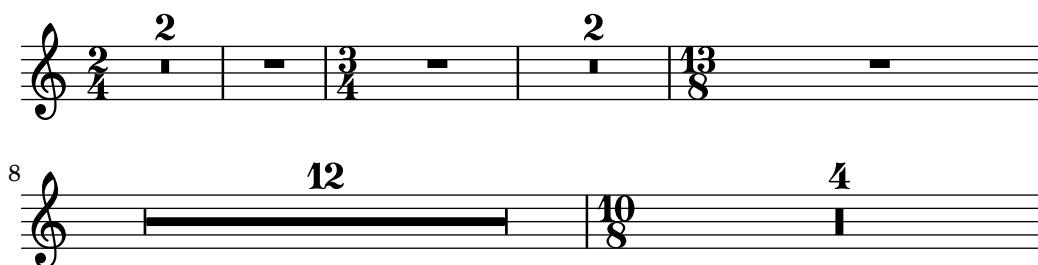
The duration of full-measure rests is identical to the duration notation used for notes. The duration in a multi-measure rest must always be an integral number of measure-lengths, so augmentation dots or fractions must often be used:

```
\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 |
}

```



A full-measure rest is printed as either a whole or breve rest, centered in the measure, depending on the time signature.

```

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

```

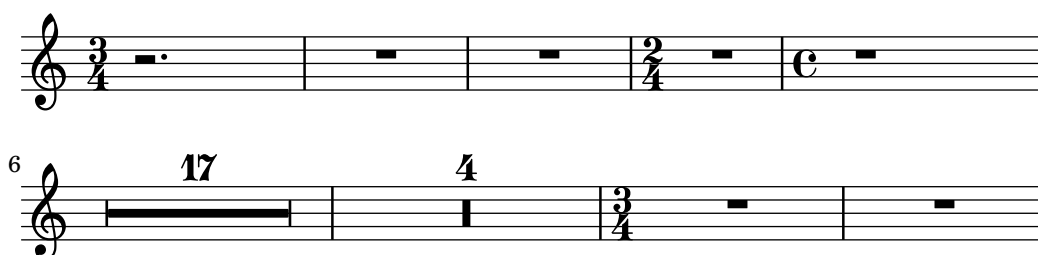


By default a multi-measure rest is expanded in the printed score to show all the rest measures explicitly. Alternatively, a multi-measure rest can be shown as a single measure containing a multi-measure rest symbol, with the number of measures of rest printed above the measure:

```

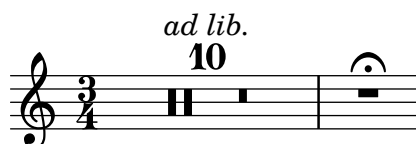
% Default behavior
\time 3/4 r2. | R2.*2 |
\time 2/4 R2 |
\time 4/4
% Rest measures contracted to single measure
\compressMMRests {
  r1 | R1*17 | R1*4 |
}
% Rest measures expanded again
\time 3/4
R2.*2 |

```



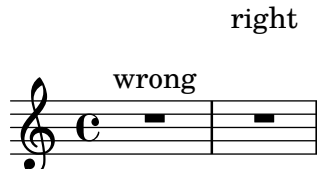
Markups can be added to multi-measure rests. The predefined command `\fermataMarkup` is provided for adding fermatas.

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



Nota: Markups attached to a multi-measure rest are objects of type `MultiMeasureRestText`, not `TextScript`. Overrides must be directed to the correct object, or they will be ignored. See the following example:

```
% This fails, as the wrong object name is specified
\override TextScript.padding = #5
R1^"wrong"
% This is the correct object name to be specified
\override MultiMeasureRestText.padding = #5
R1^"right"
```



When a multi-measure rest immediately follows a `\partial` setting, resulting bar-check warnings may not be displayed.

Instruccions predefinides

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

Fragments de codi seleccionats

Changing form of multi-measure rests

If there are ten or fewer measures of rests, a series of longa and breve rests (called in German “Kirchenpausen” - church rests) is printed within the staff; otherwise a simple line is shown. This default number of ten may be changed by overriding the `expand-limit` property.

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





Positioning multi-measure rests

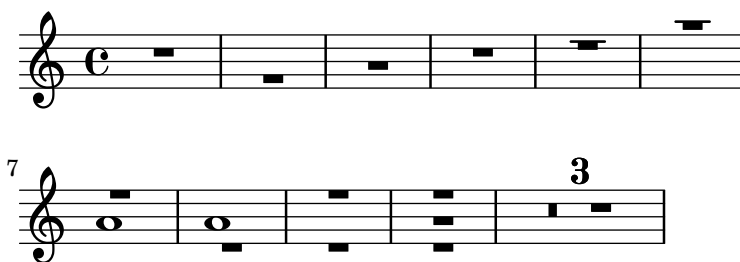
Unlike ordinary rests, there is no predefined command to change the staff position of a multi-measure rest symbol of either form by attaching it to a note. However, in polyphonic music multi-measure rests in odd-numbered and even-numbered voices are vertically separated. The positioning of multi-measure rests can be controlled as follows:

```
\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 }
  >>
}
```

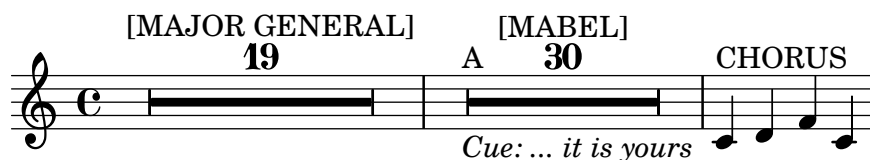


Multi-measure rest markup

Markups attached to a multi-measure rest will be centered above or below it. Long markups attached to multi-measure rests do not cause the measure to expand. To expand a multi-measure rest to fit the markup, use an empty chord with an attached markup before the multi-measure rest.

Text attached to a spacer rest in this way is left-aligned to the position where the note would be placed in the measure, but if the measure length is determined by the length of the text, the text will appear to be centered.

```
\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
  }
}
```



Vegeu també

Music Glossary: Secció “multi-measure rest” in *Glossari musical*.

Notation Reference: [Durations], pàgina 45, Secció 1.8 [Text], pàgina 229, Secció 1.8.2 [Formatting text], pàgina 237, [Text scripts], pàgina 230.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “MultiMeasureRest” in *Referència de funcionament intern*, Secció “MultiMeasureRestNumber” in *Referència de funcionament intern*, Secció “MultiMeasureRest-Text” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Fingerings over multi-measure rests (e.g., R1*10-4) may result in the fingering numeral colliding with the bar counter numeral.

There is no way to automatically condense multiple ordinary rests into a single multi-measure rest.

Multi-measure rests do not take part in rest collisions.

1.2.3 Displaying rhythms

Time signature

The time signature is set as follows:

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



Mid-measure time signature changes are covered in [Upbeats], pàgina 72.

Time signatures are printed at the beginning of a piece and whenever the time signature changes. If a change takes place at the end of a line a warning time signature sign is printed there. This default behavior may be changed, see Secció 5.4.7 [Visibility of objects], pàgina 610.

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



The time signature symbol that is used in 2/2 and 4/4 time can be changed to a numeric style:

```
\relative c'' {
  % Default style
  \time 4/4 c1
  \time 2/2 c1
  % Change to numeric style
  \numericTimeSignature
  \time 4/4 c1
  \time 2/2 c1
  % Revert to default style
  \defaultTimeSignature
  \time 4/4 c1
  \time 2/2 c1
}
```



Mensural time signatures are covered in [Mensural time signatures], pàgina 430.

In addition to setting the printed time signature, the `\time` command also sets the values of the time-signature-based properties `baseMoment`, `beatStructure`, and `beamExceptions`. The predefined default values for these properties can be found in `scm/time-signature-settings.scm`.

The default value of `beatStructure` can be overridden in the `\time` command itself by supplying it as the optional first argument:

```
\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 } |
    }
  }
}
```



Alternatively, the default values of all these time-signature-based variables, including `baseMoment` and `beamExceptions`, can be set together. The values can be set independently for several different time signatures. The new values take effect when a subsequent `\time` command with the same value of the time signature is executed:

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



`\overrideTimeSignatureSettings` takes four arguments:

1. *timeSignatureFraction*, a fraction describing the time signature to which these values apply.
2. *baseMomentFraction*, a fraction containing the numerator and denominator of the basic timing unit for the time signature.

3. *beatStructure*, a Scheme list indicating the structure of the beats in the measure, in units of the base moment.
4. *beamExceptions*, an alist containing any beaming rules for the time signature that go beyond ending at every beat, as described in [Setting automatic beam behavior], pàgina 84.

Changed values of default time signature properties can be restored to the original values:

```
\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 } |
  }
}
```



Different values of default time signature properties can be established for different staves by moving the *Timing_translator* and the *Default_bar_line_engraver* from the *Score* context to the *Staff* context.

```
\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"
}
}
}

```



A further method of changing these time-signature-related variables, which avoids reprinting the time signature at the time of the change, is shown in [Setting automatic beam behavior], pàgina 84.

Instruccions predefinides

`\numericTimeSignature`, `\defaultTimeSignature`.

Fragments de codi seleccionats

Time signature printing only the numerator as a number (instead of the fraction)

Sometimes, a time signature should not print the whole fraction (e.g. 7/4), but only the numerator (7 in this case). This can be easily done by using `\override Staff.TimeSignature.style = #'single-digit` to change the style permanently. By using `\revert Staff.TimeSignature.style`, this setting can be reversed. To apply the single-digit style to only one time signature, use the `\override` command and prefix it with a `\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
}
```



Vegeu també

Music Glossary: Secció “time signature” in *Glossari musical*

Notation Reference: [Mensural time signatures], pàgina 430, [Setting automatic beam behavior], pàgina 84, [Time administration], pàgina 118.

Installed Files: `scm/time-signature-settings.scm`.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “TimeSignature” in *Referència de funcionament intern*, Secció “Timing_translator” in *Referència de funcionament intern*.

Metronome marks

A basic metronome mark is simple to write:

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



Metronome marks may also be printed as a range of two numbers:

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



Tempo indications with text can be used instead:

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



Combining a metronome mark and text will automatically place the metronome mark within parentheses:

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



In general, the text can be any markup object:

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



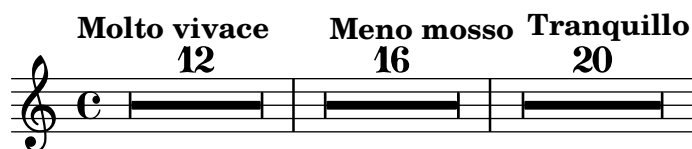
A parenthesized metronome mark with no textual indication may be written by including an empty string in the input:

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



In a part for an instrument with long periods of rests, tempo indications sometimes follow each other closely. The command `\markLengthOn` provides extra horizontal space to prevent tempo indications from overlapping, and `\markLengthOff` restores the default behavior of ignoring tempo marks for horizontal spacing.

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

Fragments de codi seleccionats

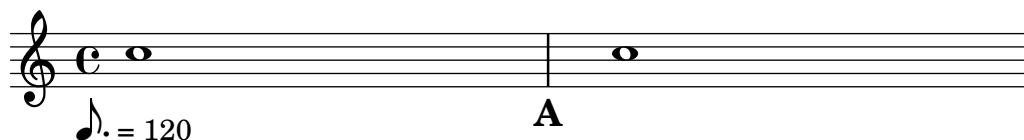
Printing metronome and rehearsal marks below the staff

By default, metronome and rehearsal marks are printed above the staff. To place them below the staff simply set the `direction` property of `MetronomeMark` or `RehearsalMark` appropriately.

```
\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
}
```



Changing the tempo without a metronome mark

To change the tempo in MIDI output without printing anything, make the metronome mark invisible.

```
\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 { }
}
```



Creating metronome marks in markup mode

New metronome marks can be created in markup mode, but they will not change the tempo in MIDI output.

```
\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
}
```



For more details, see Secció 1.8.2 [Formatting text], pàgina 237.

Vegeu també

Music Glossary: Secció “metronome” in *Glossari musical*, Secció “metronomic indication” in *Glossari musical*, Secció “tempo indication” in *Glossari musical*, Secció “metronome mark” in *Glossari musical*.

Notation Reference: Secció 1.8.2 [Formatting text], pàgina 237, Secció 3.5 [Creating MIDI output], pàgina 507.

Snippets: Secció “Staff notation” in *Fragments de codi*.

Internals Reference: Secció “MetronomeMark” in *Referència de funcionament intern*.

Upbeats

Partial or pick-up measures, such as an *anacrusis* or an *upbeat*, are entered using the `\partial` command:

```
\partial duration
```

When `\partial` is used at the beginning of a score, *duration* is the length of the music preceding the first bar.

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



When `\partial` is used after the beginning of a score, *duration* is the *remaining* length of the current measure. It does not create a new numbered bar.

```
\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 |
}
```



The `\partial` command is *required* when the time signature changes in mid measure, but it may also be used alone.

```
\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] |
}
```



The `\partial` command sets the `Timing.measurePosition` property, which is a rational number that indicates how much of the measure has passed.

Vegeu també

Music Glossary: Secció “anacrusis” in *Glossari musical*.

Notation Reference: [Grace notes], pàgina 112.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internal Reference: Secció “Timing.translator” in *Referència de funcionament intern*.

Unmetered music

In metered music bar lines are inserted and bar numbers are calculated automatically. In unmetered music (i.e., cadenzas), this is not desirable and can be ‘switched off’ using the command `\cadenzaOn`, then ‘switched back on’ at the appropriate place using `\cadenzaOff`.

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

```

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

```



Bar numbering is resumed at the end of the cadenza.

```

\relative c'' {
  % Show all bar numbers
  \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
}

```



Inserting a `\bar` command within a cadenza does not start a new measure, even if a bar line is printed. So any accidentals – which are usually assumed to remain in force until the end of the measure – will still be valid after the bar line printed by `\bar`. If subsequent accidentals should be printed, forced accidentals or reminder accidentals need to be inserted manually, see [\[Accidentals\]](#), pàgina [\[Accidentals\]](#).

```

\relative c'' {
  c4 d e d
  \cadenzaOn
  cis4 d cis d
  \bar "|"
  % First cis is printed without alteration even if it's after a \bar
  cis4 d cis! d
  \cadenzaOff
  \bar "|"
}

```



Automatic beaming is disabled by `\cadenzaOn`. Therefore, all beaming in cadenzas must be entered manually. See [\[Manual beams\]](#), pàgina 93.

```

\relative {
  \repeat unfold 8 { c''8 }
}

```

```

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

```



These predefined commands affect all staves in the score, even when placed in just one `Voice` context. To change this, move the `Timing_translator` from the `Score` context to the `Staff` context. See [Polymetric notation], pàgina 75.

Instruccions predefinides

`\cadenzaOn`, `\cadenzaOff`.

Vegeu també

Music Glossary: Secció “cadenza” in *Glossari musical*.

Notation Reference: Secció 5.4.7 [Visibility of objects], pàgina 610, [Polymetric notation], pàgina 75, [Manual beams], pàgina 93, <undefined> [Accidentals], pàgina <undefined>.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Advertiments i problemes coneguts

Automatic line and page breaks are inserted only at bar lines, so ‘invisible’ bar lines will need to be inserted manually in long stretches of unmeasured music to permit breaking:

```
\bar ""
```

Polymetric notation

Polymetric notation is supported explicitly or by manually modifying the visible time signature symbol and/or scaling note durations.

Different time signatures with equal-length measures

Set a common time signature for each staff, and set the `timeSignatureFraction` to the desired fraction. Then use the `\scaleDurations` function to scale the durations of the notes in each staff to the common time signature.

In the following example, music with the time signatures of 3/4, 9/8 and 10/8 are used in parallel. In the second staff, shown durations are multiplied by 2/3 (because $2/3 * 9/8 = 3/4$) and in the third staff, the shown durations are multiplied by 3/5 (because $3/5 * 10/8 = 3/4$). It may be necessary to insert beams manually, as the duration scaling will affect the autobeaming rules.

```

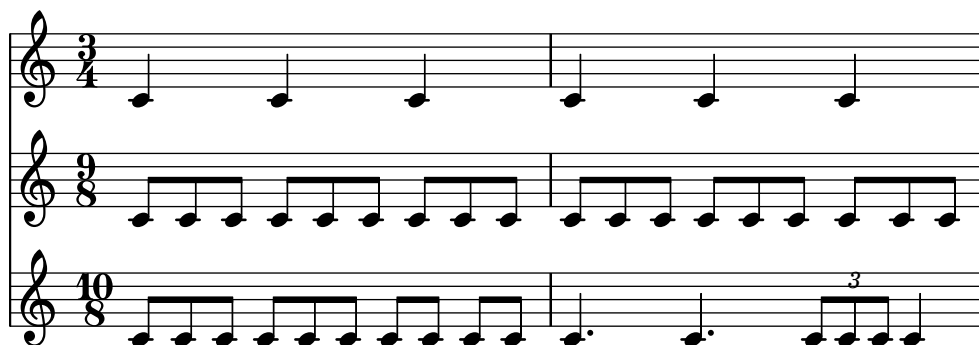
\relative <<
\new Staff {
  \time 3/4
  c'4 c c |
  c4 c c |
}
\new Staff {

```

```

\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
  }
}
>>

```



Different time signatures with unequal-length measures

Each staff can be given its own independent time signature by moving the `Timing_translator` and the `Default_bar_line_engraver` to the `Staff` context.

```

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

% Now each staff has its own time signature.

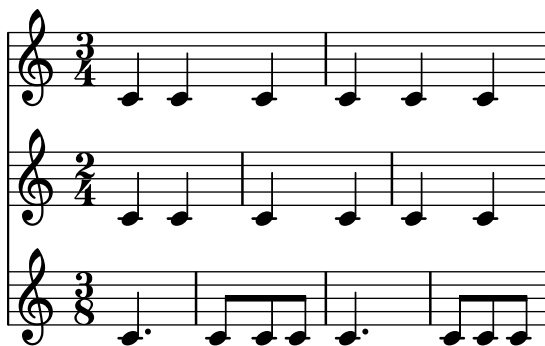
\relative <<
\new Staff {
  \time 3/4
  c'4 c c |
  c4 c c |
}
\new Staff {

```

```

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

```



Compound time signatures

These are created using the `\compoundMeter` function. The syntax for this is:

```
\compoundMeter #'(list of lists)
```

The simplest construction is a single list, where the *last* number indicates the bottom number of the time signature and those that come before it, the top numbers.

```

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

```



More complex meters can be constructed using additional lists. Also, automatic beaming settings will be adjusted depending on the values.

```

\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
}

```



Vegeu també

Music Glossary: Secció “polymetric” in *Glossari musical*, Secció “polymetric time signature” in *Glossari musical*, Secció “meter” in *Glossari musical*.

Notation Reference: [Automatic beams], pàgina 82, [Manual beams], pàgina 93, [Time signature], pàgina 65, [Scaling durations], pàgina 52.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “TimeSignature” in *Referència de funcionament intern*, Secció “Timing_translator” in *Referència de funcionament intern*, Secció “Default_bar_line_engraver” in *Referència de funcionament intern*, Secció “Staff” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

When using different time signatures in parallel, notes at the same moment will be placed at the same horizontal location. However, the bar lines in the different staves will cause the note spacing to be less regular in each of the individual staves than would be normal without the different time signatures.

Automatic note splitting

Long notes which overrun bar lines can be converted automatically to tied notes. This is done by replacing the `Note_heads_engraver` with the `Completion_heads_engraver`. Similarly, long rests which overrun bar lines are split automatically by replacing the `Rest_engraver` with the `Completion_rest_engraver`. In the following example, notes and rests crossing the bar lines are split, notes are also tied.

```
\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
}
```



These engravers split all running notes and rests at the bar line, and inserts ties for notes. One of its uses is to debug complex scores: if the measures are not entirely filled, then the ties show exactly how much each measure is off.

The property `completionUnit` sets a preferred duration for the split notes.

```
\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.
}

```



These engravers split notes with scaled duration, such as those in tuplets, into notes with the same scale-factor as in the input note.

```

\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
}

```



Vegeu també

Music Glossary: Secció “tie” in *Glossari musical*

Learning Manual: Secció “Engravers explained” in *Manual d’aprenentatge*, Secció “Adding and removing engravers” in *Manual d’aprenentatge*.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “Note_heads_engraver” in *Referència de funcionament intern*, Secció “Completion_heads_engraver” in *Referència de funcionament intern*, Secció “Rest_engraver” in *Referència de funcionament intern*, Secció “Completion_rest_engraver” in *Referència de funcionament intern*, Secció “Forbid_line_break_engraver” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

For consistency with previous behavior, notes and rests with duration longer than a measure, such as `c1*2`, are split into notes without any scale factor, `{ c1 c1 }`. The property `completionFactor` controls this behavior, and setting it to `#f` cause split notes and rests to have the scale factor of the input durations.

Showing melody rhythms

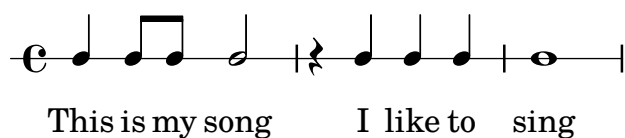
Sometimes you might want to show only the rhythm of a melody. This can be done with the rhythmic staff. All pitches of notes on such a staff are squashed, and the staff itself has a single line

<<

```

\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
  }
}
>>

```

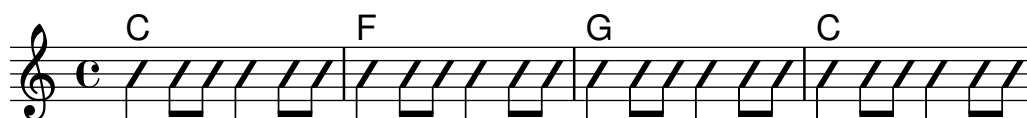


Guitar chord charts often show the strumming rhythms. This can be done with the `Pitch_squash_engraver` and `\improvisationOn`.

```

<<
\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
}
>>

```



Instruccions predefinides

`\improvisationOn`, `\improvisationOff`.

Fragments de codi seleccionats

Guitar strum rhythms

For guitar music, it is possible to show strum rhythms, along with melody notes, chord names and fret diagrams.

```
\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.
    }
  }
>>

```

The image displays a musical score for a song. At the top, three guitar chord diagrams are shown: C (C major), F (F major), and G (G major). Below these, a musical staff is presented with a treble clef and a common time signature (C). The staff contains a melody line with notes and rests, and a guitar accompaniment line with chords. The lyrics 'This is my song. I like' are written below the staff.

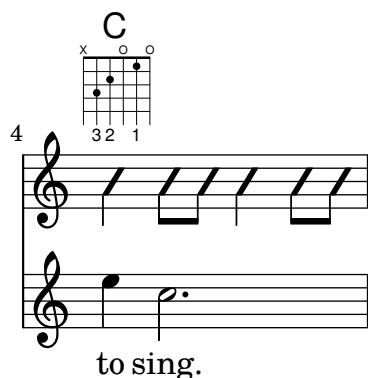
The chord diagrams are as follows:

- C (C major):** x (open), 0 (open), 0 (open), 2 (second fret), 3 (third fret), 1 (first fret).
- F (F major):** 1 (first fret), 3 (third fret), 4 (fourth fret), 2 (second fret), 1 (first fret), 1 (first fret).
- G (G major):** 2 (second fret), 1 (first fret), 3 (third fret), 0 (open), 0 (open), 0 (open).

The musical staff shows the following notes and rests:

- Measure 1: C4 (quarter), E4 (quarter), E4 (quarter).
- Measure 2: F2 (half), rest (half).
- Measure 3: G2 (half), A4 (quarter).
- Measure 4: E4 (quarter), C2 (half).

The lyrics are: This is my song. I like



Vegeu també

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “RhythmicStaff” in *Referència de funcionament intern*, Secció “Pitch_squash_engraver” in *Referència de funcionament intern*.

1.2.4 Beams

Automatic beams

By default, beams are inserted automatically:

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



If these automatic decisions are not satisfactory, beaming can be entered explicitly; see [Manual beams], pàgina 93. Beams *must* be entered manually if beams are to be extended over rests.

If automatic beaming is not required, it may be turned off with `\autoBeamOff` and on with `\autoBeamOn`:

```
\relative c' {
  c4 c8 c8. c16 c8. c16 c8
  \autoBeamOff
  c4 c8 c8. c16 c8.
  \autoBeamOn
  c16 c8
}
```



Nota: If beams are used to indicate melismata in songs, then automatic beaming should be switched off with `\autoBeamOff` and the beams indicated manually. Using `\partcombine` with `\autoBeamOff` can produce unintended results. See the snippets for more information.

Beaming patterns that differ from the automatic defaults can be created; see [Setting automatic beam behavior], pàgina 84.

Instruccions predefinides

`\autoBeamOff`, `\autoBeamOn`.

Fragments de codi seleccionats

Beams across line breaks

Line breaks are normally forbidden when beams cross bar lines. This behavior can be changed as shown:

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



Changing beam knee gap

Kneed beams are inserted automatically when a large gap is detected between the note heads. This behavior can be tuned through the `auto-knee-gap` property. A kneed beam is drawn if the gap is larger than the value of `auto-knee-gap` plus the width of the beam object (which depends on the duration of the notes and the slope of the beam). By default `auto-knee-gap` is set to 5.5 staff spaces.

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



Partcombine and autoBeamOff

The function of `\autoBeamOff` when used with `\partcombine` can be difficult to understand.

It may be preferable to use

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

instead, to ensure that autobeaming will be turned off for the entire staff.

`\partcombine` apparently works with 3 voices – stem up single, stem down single, stem up combined.

An `\autoBeamOff` call in the first argument to `partcombine` will apply to the voice that is active at the time the call is processed, either stem up single or stem up combined. An `\autoBeamOff` call in the second argument will apply to the voice that is stem down single.

In order to use `\autoBeamOff` to stop all autobeaming when used with `\partcombine`, it will be necessary to use three calls to `\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 |
  }
}
```



Vegeu també

Notation Reference: [Manual beams], pàgina 93, [Setting automatic beam behavior], pàgina 84.

Installed Files: `scm/auto-beam.scm`.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “Auto_beam_engraver” in *Referència de funcionament intern*, Secció “Beam_engraver” in *Referència de funcionament intern*, Secció “Beam” in *Referència de funcionament intern*, Secció “BeamEvent” in *Referència de funcionament intern*, Secció “Beam-ForbidEvent” in *Referència de funcionament intern*, Secció “beam-interface” in *Referència de funcionament intern*, Secció “unbreakable-spanner-interface” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

The properties of a beam are determined at the *start* of its construction and any additional beam-property changes that occur before the beam has been completed will not take effect until the *next*, new beam starts.

Setting automatic beam behavior

When automatic beaming is enabled, the placement of automatic beams is determined by three context properties: `baseMoment`, `beatStructure`, and `beamExceptions`. The default values of these variables may be overridden as described below, or alternatively the default values themselves may be changed as explained in [Time signature], pàgina 65.

If a `beamExceptions` rule is defined for the time signature in force, that rule alone is used to determine the beam placement; the values of `baseMoment` and `beatStructure` are ignored.

If no `beamExceptions` rule is defined for the time signature in force, the beam placement is determined by the values of `baseMoment` and `beatStructure`.

Beaming based on baseMoment and beatStructure

By default, `beamExceptions` rules are defined for most common time signatures, so the `beamExceptions` rules must be disabled if automatic beaming is to be based on `baseMoment` and `beatStructure`. The `beamExceptions` rules are disabled by

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

When `beamExceptions` is set to `#'()`, either due to an explicit setting or because no `beamExceptions` rules are defined internally for the time signature in force, the ending points for beams are on beats as specified by the context properties `baseMoment` and `beatStructure`. `beatStructure` is a scheme list that defines the length of each beat in the measure in units of `baseMoment`. By default, `baseMoment` is one over the denominator of the time signature. By default, each unit of length `baseMoment` is a single beat.

Note that there are separate `beatStructure` and `baseMoment` values for each time signature. Changes to these variables apply only to the time signature that is currently in force, hence those changes must be placed after the `\time` command which starts a new time signature section, not before it. New values given to a particular time signature are retained and reinstated whenever that time signature is re-established.

```
\relative c'' {
  \time 5/16
  c16^"default" c c c c |
  % beamExceptions are unlikely to be defined for 5/16 time,
  % but let's disable them anyway to be sure
  \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^"default" a a a a a a
  % Disable beamExceptions because they are definitely
  % defined for 4/4 time
  \set Timing.beamExceptions = #'()
  \set Timing.baseMoment = #(ly:make-moment 1/4)
  \set Timing.beatStructure = 1,1,1,1
  a8^"changed" a a a a a a
}
```



Beam setting changes can be limited to specific contexts. If no setting is included in a lower-level context, the setting of the enclosing context will apply.

```
\new Staff {
```

```

\time 7/8
% No need to disable beamExceptions
% as they are not defined for 7/8 time
\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
    }
  }
>>
}

```



When multiple voices are used the `Staff` context must be specified if the beaming is to be applied to all voices in the staff:

```

\time 7/8
% rhythm 3-1-1-2
% Change applied to Voice by default -- does not work correctly
% Because of autogenerated voices, all beaming will
% be at 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} >>

% Works correctly with context Staff specified
\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} >>

```



The value of `baseMoment` can be adjusted to change the beaming behavior, if desired. When this is done, the value of `beatStructure` must be set to be compatible with the new value of `baseMoment`.

```

\time 5/8
% No need to disable beamExceptions
% as they are not defined for 5/8 time
\set Timing.baseMoment = #(ly:make-moment 1/16)
\set Timing.beatStructure = 7,3
\repeat unfold 10 { a'16 }

```




`baseMoment` is a *moment*; a unit of musical duration. A quantity of type *moment* is created by the scheme function `ly:make-moment`. For more information about this function, see [Time administration], pàgina 118.

By default `baseMoment` is set to one over the denominator of the time signature. Any exceptions to this default can be found in `scm/time-signature-settings.scm`.

Beaming based on `beamExceptions`

Special autobeaaming rules (other than ending a beam on a beat) are defined in the `beamExceptions` property.

The value for `beamExceptions`, a somewhat complex Scheme data structure, is easiest generated with the `\beamExceptions` function. This function is given one or more manually beamed measure-length rhythmic patterns (measures have to be separated by a bar check `|` since the function has no other way to discern the measure length). Here is a simple example:

```
\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: A `beamExceptions` value must be *complete* exceptions list. That is, every exception that should be applied must be included in the setting. It is not possible to add, remove, or change only one of the exceptions. While this may seem cumbersome, it means that the current beaming settings need not be known in order to specify a new beaming pattern.

When the time signature is changed, default values of `Timing.baseMoment`, `Timing.beatStructure`, and `Timing.beamExceptions` are set. Setting the time signature will reset the automatic beaming settings for the `Timing` context to the default behavior.

```
\relative a' {
  \time 6/8
  \repeat unfold 6 { a8 }
  % group (4 + 2)
  \set Timing.beatStructure = 4,2
  \repeat unfold 6 { a8 }
  % go back to default behavior
  \time 6/8
  \repeat unfold 6 { a8 }
}
```



The default automatic beaming settings for a time signature are determined in `scm/time-signature-settings.scm`. Changing the default automatic beaming settings for a time signature is described in [Time signature], pàgina 65.

Many automatic beaming settings for a time signature contain an entry for `beamExceptions`. For example, 4/4 time tries to beam the measure in two if there are only eighth notes. The `beamExceptions` rule can override the `beatStructure` setting if `beamExceptions` is not reset.

```
\time 4/4
\set Timing.baseMoment = #(ly:make-moment 1/8)
\set Timing.beatStructure = 3,3,2
% This won't beam (3 3 2) because of beamExceptions
\repeat unfold 8 {c''8} |
% This will beam (3 3 2) because we clear beamExceptions
\set Timing.beamExceptions = #'()
\repeat unfold 8 {c''8}
```



In a similar fashion, eighth notes in 3/4 time are beamed as a full measure by default. To beam eighth notes in 3/4 time on the beat, reset `beamExceptions`.

```
\time 3/4
% by default we beam in (6) due to beamExceptions
\repeat unfold 6 {a'8} |
% This will beam (1 1 1) due to default baseMoment and beatStructure
\set Timing.beamExceptions = #'()
\repeat unfold 6 {a'8}
```



In engraving from the Romantic and Classical periods, beams often begin midway through the measure in 3/4 time, but modern practice is to avoid the false impression of 6/8 time (see Gould, p. 153). Similar situations arise in 3/8 time. This behavior is controlled by the context property `beamHalfMeasure`, which has effect only in time signatures with 3 in the numerator:

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



How automatic beaming works

When automatic beaming is enabled, the placement of automatic beams is determined by the context properties `baseMoment`, `beatStructure`, and `beamExceptions`.

The following rules, in order of priority, apply when determining the appearance of beams:

- If a manual beam is specified with [...] set the beam as specified, otherwise
- if a beam-ending rule is defined in `beamExceptions` for the beam-type, use it to determine the valid places where beams may end, otherwise
- if a beam-ending rule is defined in `beamExceptions` for a longer beam-type, use it to determine the valid places where beams may end, otherwise
- use the values of `baseMoment` and `beatStructure` to determine the ends of the beats in the measure, and end beams at the end of beats.

In the rules above, the *beam-type* is the duration of the shortest note in the beamed group.

The default beaming rules can be found in `scm/time-signature-settings.scm`.

Fragments de codi seleccionats

Subdividing beams

The beams of consecutive 16th (or shorter) notes are, by default, not subdivided. That is, the three (or more) beams stretch unbroken over entire groups of notes. This behavior can be modified to subdivide the beams into sub-groups by setting the property `subdivideBeams`. When set, multiple beams will be subdivided at intervals defined by the current value of `baseMoment` by reducing the multiple beams to the number of beams that indicates the metric value of the subdivision. If the group following the division is shorter than the current metric value (usually because the beam is incomplete) the number of beams reflects the longest possible subdivision group. However, if there is only one note left after the division this restriction isn't applied. Note that `baseMoment` defaults to one over the denominator of the current time signature if not set explicitly. It must be set to a fraction giving the duration of the beam sub-group using the `ly:make-moment` function, as shown in this snippet. Also, when `baseMoment` is changed, `beatStructure` should also be changed to match the new `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
}

```



Strict beat beaming

Beamlets can be set to point in the direction of the beat to which they belong. The first beam avoids sticking out flags (the default); the second beam strictly follows the beat.

```

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

```



Conducting signs measure grouping signs

Beat grouping within a measure is controlled by the context property `beatStructure`. Values of `beatStructure` are established for many time signatures in `scm/time-signature-settings.scm`. Values of `beatStructure` can be changed or set with `\set`. Alternatively, `\time` can be used to both set the time signature and establish the beat structure. For this, you specify the internal grouping of beats in a measure as a list of numbers (in Scheme syntax) before the time signature.

`\time` applies to the `Timing` context, so it will not reset values of `beatStructure` or `baseMoment` that are set in other lower-level contexts, such as `Voice`.

If the `Measure_grouping_engraver` is included in one of the display contexts, measure grouping signs will be created. Such signs ease reading rhythmically complex modern music. In the example, the 9/8 measure is grouped in two different patterns using the two different methods, while the 5/8 measure is grouped according to the default setting in `scm/time-signature-settings.scm`:

```

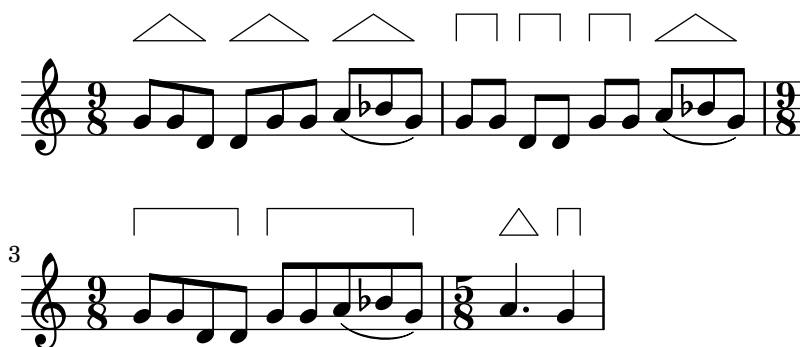
\score {
  \new Voice \relative c'' {
    \time 9/8
    g8 g d d g g a( bes g) |
    \set Timing.beatStructure = 2,2,2,3
    g8 g d d g g a( bes g) |
    \time 4,5 9/8
    g8 g d d g g a( bes g) |
    \time 5/8

```

```

    a4. g4 |
  }
  \layout {
    \context {
      \Staff
      \consists "Measure_grouping_engraver"
    }
  }
}

```



Beam endings in Score context

Beam-ending rules specified in the **Score** context apply to all staves, but can be modified at both **Staff** and **Voice** levels:

```

\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
        }
      >>
    }
  >>
}

```

```

    }
  >>
}

```



Vegeu també

Notation Reference: [Time signature], pàgina 65.

Installed Files: `scm/time-signature-settings.scm`.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “Auto_beam_engraver” in *Referència de funcionament intern*, Secció “Beam” in *Referència de funcionament intern*, Secció “BeamForbidEvent” in *Referència de funcionament intern*, Secció “beam-interface” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

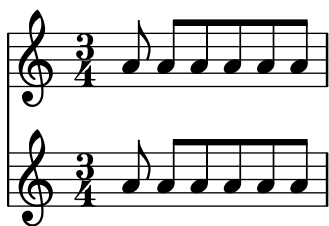
If a score ends while an automatic beam has not been ended and is still accepting notes, this last beam will not be typeset at all. The same holds for polyphonic voices, entered with `<< ... \\ ... >>`. If a polyphonic voice ends while an automatic beam is still accepting notes, it is not typeset. The workaround for these problems is to manually beam the last beam in the voice or score.

By default, the `Timing` translator is aliased to the `Score` context. This means that setting the time signature in one staff will affect the beaming of the other staves as well. Thus, a time signature setting in a later staff will reset custom beaming that was set in an earlier staff. One way to avoid this problem is to set the time signature in only one staff.

```

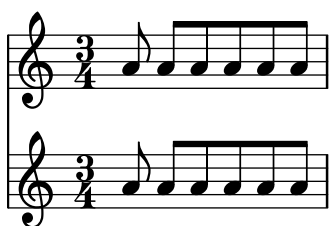
<<
  \new Staff {
    \time 3/4
    \set Timing.baseMoment = #(ly:make-moment 1/8)
    \set Timing.beatStructure = 1,5
    \set Timing.beamExceptions = #'()
    \repeat unfold 6 { a'8 }
  }
  \new Staff {
    \repeat unfold 6 { a'8 }
  }
>>

```



The default beam settings for the time signature can also be changed, so that the desired beaming will always be used. Changes in automatic beaming settings for a time signature are described in [Time signature], pàgina 65.

```
<<
\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 }
}
>>
```



Manual beams

In some cases it may be necessary to override the automatic beaming algorithm. For example, the autobeamer will not put beams over rests or bar lines, and in choral scores the beaming is often set to follow the meter of the lyrics rather than the notes. Such beams can be specified manually by marking the begin and end point with [and].

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



Beaming direction can be set manually using direction indicators:

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



Individual notes may be marked with `\noBeam` to prevent them from being beamed:

```
\relative {
```

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



Grace note beams and normal note beams can occur simultaneously. Unbeamed grace notes are not put into normal note beams.

```
\relative {
  c''4 d8[
    \grace { e32 d c d }
  e8] e[ e
    \grace { f16 }
  e8 e]
}
```



Even more strict manual control with the beams can be achieved by setting the properties `stemLeftBeamCount` and `stemRightBeamCount`. They specify the number of beams to draw on the left and right side, respectively, of the next note. If either property is set, its value will be used only once, and then it is erased. In this example, the last `f` is printed with only one beam on the left side, i.e., the eighth-note beam of the group as a whole.

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



Instruccions predefinides

`\noBeam.`

Fragments de codi seleccionats

Flat flags and beam nibs

Flat flags on lone notes and beam nibs at the ends of beamed figures are both possible with a combination of `stemLeftBeamCount`, `stemRightBeamCount` and paired `[]` beam indicators.

For right-pointing flat flags on lone notes, use paired `[]` beam indicators and set `stemLeftBeamCount` to zero (see Example 1).

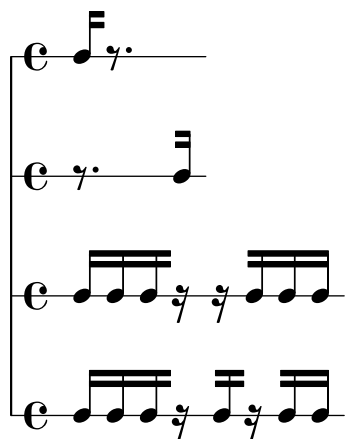
For left-pointing flat flags, set `stemRightBeamCount` instead (Example 2).

For right-pointing nibs at the end of a run of beamed notes, set `stemRightBeamCount` to a positive value. And for left-pointing nibs at the start of a run of beamed notes, set `stemLeftBeamCount` instead (Example 3).

Sometimes it may make sense for a lone note surrounded by rests to carry both a left- and right-pointing flat flag. Do this with paired `[]` beam indicators alone (Example 4).

(Note that `\set stemLeftBeamCount` is always equivalent to `\once \set`. In other words, the beam count settings are not “sticky”, so the pair of flat flags attached to the lone `16[]` in the last example have nothing to do with the `\set` two notes prior.)

```
\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
    }
  >>
}
```



Vegeu també

Notation Reference: Secció 5.4.2 [Direction and placement], pàgina 602, [Grace notes], pàgina 112.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “Beam” in *Referència de funcionament intern*, Secció “BeamEvent” in *Referència de funcionament intern*, Secció “Beam_engraver” in *Referència de funcionament intern*, Secció “beam-interface” in *Referència de funcionament intern*, Secció “Stem_engraver” in *Referència de funcionament intern*.

Feathered beams

Feathered beams are used to indicate that a small group of notes should be played at an increasing (or decreasing) tempo, without changing the overall tempo of the piece. The extent of the feathered beam must be indicated manually using [and], and the beam feathering is turned on by specifying a direction to the `Beam` property `grow-direction`.

If the placement of the notes and the sound in the MIDI output is to reflect the *ritardando* or *accelerando* indicated by the feathered beam the notes must be grouped as a music expression delimited by braces and preceded by a `featherDurations` command which specifies the ratio between the durations of the first and last notes in the group.

The square brackets show the extent of the beam and the braces show which notes are to have their durations modified. Normally these would delimit the same group of notes, but this is not required: the two commands are independent.

In the following example the eight 16th notes occupy exactly the same time as a half note, but the first note is one half as long as the last one, with the intermediate notes gradually lengthening. The first four 32nd notes gradually speed up, while the last four 32nd notes are at a constant 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] }
  % revert to non-feathered beams
  \override Beam.grow-direction = #'()
  { g32[ a b c] }
}
```



The spacing in the printed output represents the note durations only approximately, but the MIDI output is exact.

Instruccions predefinides

`\featherDurations`.

Vegeu també

Snippets: Secció “Rhythms” in *Fragments de codi*.

Advertiments i problemes coneguts

The `\featherDurations` command only works with very short music snippets, and when numbers in the fraction are small.

1.2.5 Bars

Bar lines

Bar lines delimit measures, and are also used to indicate repeats. Normally, simple bar lines are automatically inserted into the printed output at places based on the current time signature.

The simple bar lines inserted automatically can be changed to other types with the `\bar` command. For example, a closing double bar line is usually placed at the end of a piece:

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



It is not invalid if the final note in a measure does not end on the automatically entered bar line: the note is assumed to carry over into the next measure. But if a long sequence of such carry-over measures appears the music can appear compressed or even flowing off the page. This is because automatic line breaks happen only at the end of complete measures, i.e., where all notes end before the end of a measure.

Nota: An incorrect duration can cause line breaks to be inhibited, leading to a line of highly compressed music or music that flows off the page.

Line breaks are also permitted at manually inserted bar lines even within incomplete measures. To allow a line break without printing a bar line, use the following:

```
\bar ""
```

This will insert an invisible bar line and allow (but not force) a line break to occur at this point. The bar number counter is not increased. To force a line break see Secció 4.3.1 [Line breaking], pàgina 535.

This and other special bar lines may be inserted manually at any point. When they coincide with the end of a measure they replace the simple bar line which would have been inserted there automatically. When they do not coincide with the end of a measure the specified bar line is inserted at that point in the printed output.

Note that manual bar lines are purely visual. They do not affect any of the properties that a normal bar line would affect, such as measure numbers, accidentals, line breaks, etc. They do not affect the calculation and placement of subsequent automatic bar lines. When a manual bar

line is placed where a normal bar line already exists, the effects of the original bar line are not altered.

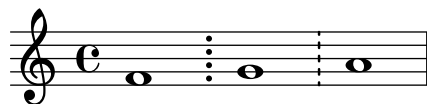
Two types of simple bar lines and five types of double bar lines are available for manual insertion:

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



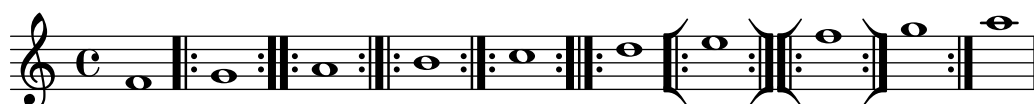
together with dotted and dashed bar lines:

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



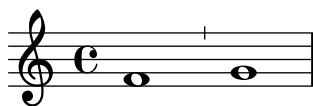
and nine types of repeat bar lines:

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



Additionally, a bar line can be printed as a simple tick:

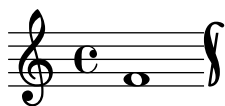
```
f'1 \bar "'" g'1
```



However, as such ticks are typically used in Gregorian chant, it is preferable to use `\divisioMinima` there instead, described in the section [Divisiones], pàgina 437, in Gregorian chant.

LilyPond supports kievian notation and provides a special kievian bar line:

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



Further details of this notation are explained in Secció 2.9.5 [Typesetting Kievan square notation], pàgina 446.

For in-line segno signs, there are three types of bar lines which differ in their behavior at line breaks:

```
\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
}
```

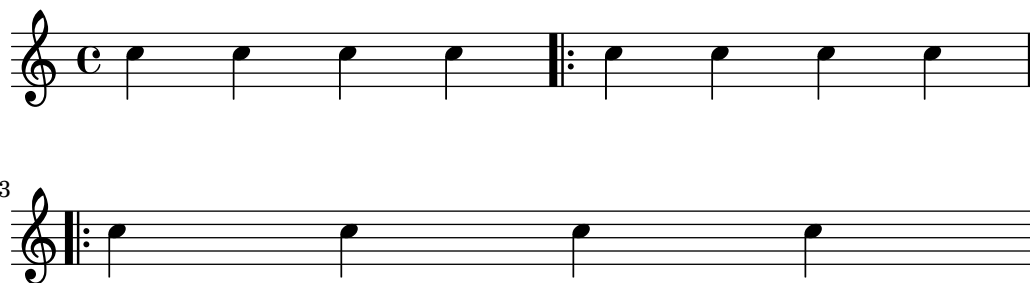


Although the bar line types signifying repeats may be inserted manually they do not in themselves cause LilyPond to recognize a repeated section. Such repeated sections are better entered

using the various repeat commands (see Secció 1.4 [Repeats], pàgina 147), which automatically print the appropriate bar lines.

In addition, you can specify ".|:-||", which is equivalent to ".|:" except at line breaks, where it gives a double bar line at the end of the line and a start repeat at the beginning of the next line.

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



For combinations of repeats with the segno sign, there are six different variations:

```
\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
}
```



Additionally there is an `\inStaffSegno` command which creates a segno bar line in conjunction with an appropriate repeat bar line when used with a `\repeat volta` command, see [Normal repeats], pàgina 148.

New bar line types can be defined with `\defineBarLine`:

```
\defineBarLine bartype #'(end begin span)
```

The `\defineBarline` variables can include the ‘empty’ string `""`, which is equivalent to an invisible bar line being printed. Or they can be set to `#f` which prints no bar line at all.

After the definiton, the new bar line can be used by `\bar bartype`.

There are currently ten bar line elements available:

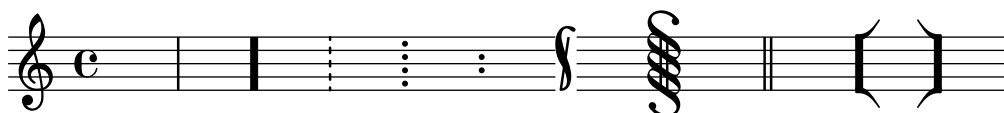
```
\defineBarLine ":" #'(" " ":" " ")
\defineBarLine "=" #'("=" " " " ")
\defineBarLine "[" #'(" " "[" " ")
\defineBarLine "]" #'("]" " " " ")
```

```
\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 ""
}

```



The "=" bar line provides the double span bar line, used in combination with the segno sign. Do not use it as a standalone double thin bar line; here, `\bar "||"` is preferred.

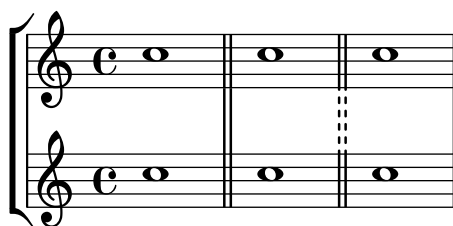
The "-" sign starts annotations to bar lines which are useful to distinguish those with identical appearance but different behavior at line breaks and/or different span bars. The part following the "-" sign is not used for building up the bar line.

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

```

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

```



Furthermore, the space character " " serves as a placeholder for defining span bars correctly aligned to the main bar lines:

```

\defineBarLine " :|.-wrong" #'(" :|." "" " |.")
\defineBarLine " :|.-right" #'(" :|." "" " |.")

\new StaffGroup <<
  \new Staff \relative c'' {
    c1 \bar " :|.-wrong"
    c1 \bar " :|.-right"
    c1
  }
  \new Staff \relative c'' {
    c1
  }
>>

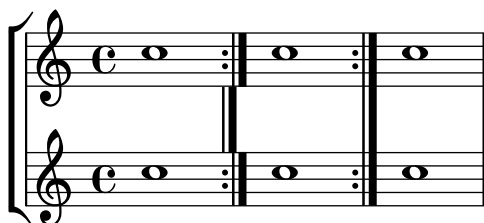
```



```

c1
c1
}
>>

```



If additional elements are needed, LilyPond provides a simple way to define them. For more information on modifying or adding bar lines, see file `scm/bar-line.scm`.

In scores with many staves, a `\bar` command in one staff is automatically applied to all staves. The resulting bar lines are connected between different staves of a `StaffGroup`, `PianoStaff`, or `GrandStaff`.

```

<<
  \new StaffGroup <<
    \new Staff \relative {
      e'4 d
      \bar "||"
      f4 e
    }
    \new Staff \relative { \clef bass c'4 g e g }
  >>
  \new Staff \relative { \clef bass c'2 c2 }
>>

```



The command `\bar bartype` is a shortcut for `\set Timing.whichBar = bartype`. A bar line is created whenever the `whichBar` property is set.

The default bar type used for automatically inserted bar lines is `"|"`. This may be changed at any time with `\set Timing.defaultBarType = bartype`.

Vegeu també

Notation Reference: Secció 4.3.1 [Line breaking], pàgina 535, Secció 1.4 [Repeats], pàgina 147, [Grouping staves], pàgina 187.

Installed Files: `scm/bar-line.scm`.

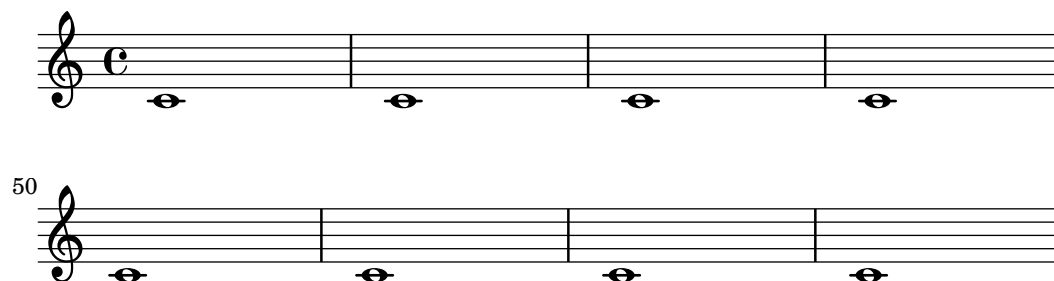
Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “BarLine” in *Referència de funcionament intern* (created at **Staff** level), Secció “SpanBar” in *Referència de funcionament intern* (across staves), Secció “Timing_translator” in *Referència de funcionament intern* (for Timing properties).

Bar numbers

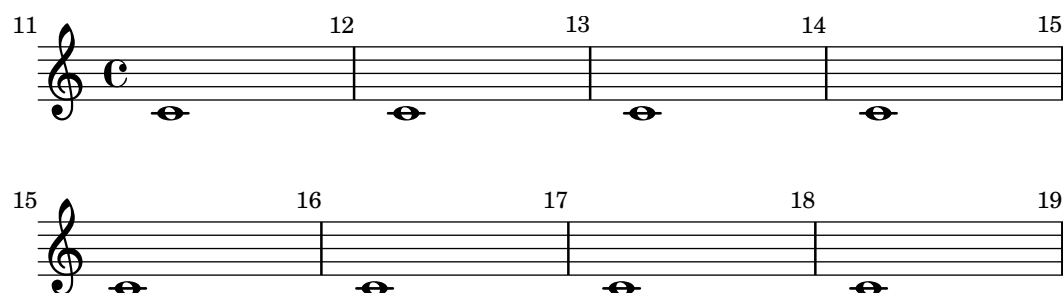
Bar numbers are typeset by default at the start of every line except the first line. The number itself is stored in the `currentBarNumber` property, which is normally updated automatically for every measure. It may also be set manually:

```
\relative c' {
  c1 c c c
  \break
  \set Score.currentBarNumber = #50
  c1 c c c
}
```



Bar numbers can be typeset at regular intervals instead of just at the beginning of every line. To do this the default behavior must be overridden to permit bar numbers to be printed at places other than the start of a line. This is controlled by the `break-visibility` property of `BarNumber`. This takes three values which may be set to `#t` or `#f` to specify whether the corresponding bar number is visible or not. The order of the three values is `end of line visible`, `middle of line visible`, `beginning of line visible`. In the following example bar numbers are printed at all possible places:

```
\relative c' {
  \override Score.BarNumber.break-visibility = ##(#t #t #t)
  \set Score.currentBarNumber = #11
  % Permit first bar number to be printed
  \bar ""
  c1 | c | c | c |
  \break
  c1 | c | c | c |
}
```



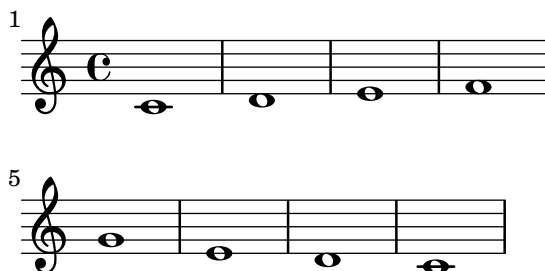
Fragments de codi seleccionats

Printing the bar number for the first measure

By default, the first bar number in a score is suppressed if it is less than or equal to ‘1’. By setting `barNumberVisibility` to `all-bar-numbers-visible`, any bar number can be printed for the first measure and all subsequent measures. Note that an empty bar line must be inserted before the first note for this to work.

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

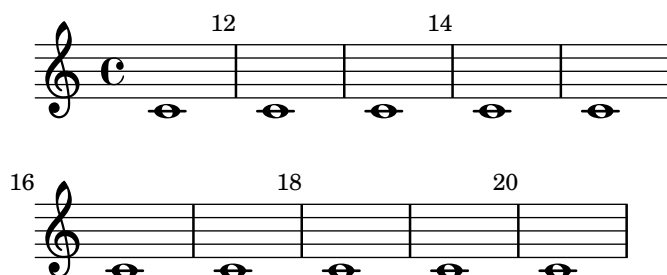
\relative c' {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  \bar ""
  c1 | d | e | f \break
  g1 | e | d | c
}
```



Printing bar numbers at regular intervals

Bar numbers can be printed at regular intervals by setting the property `barNumberVisibility`. Here the bar numbers are printed every two measures except at the end of the line.

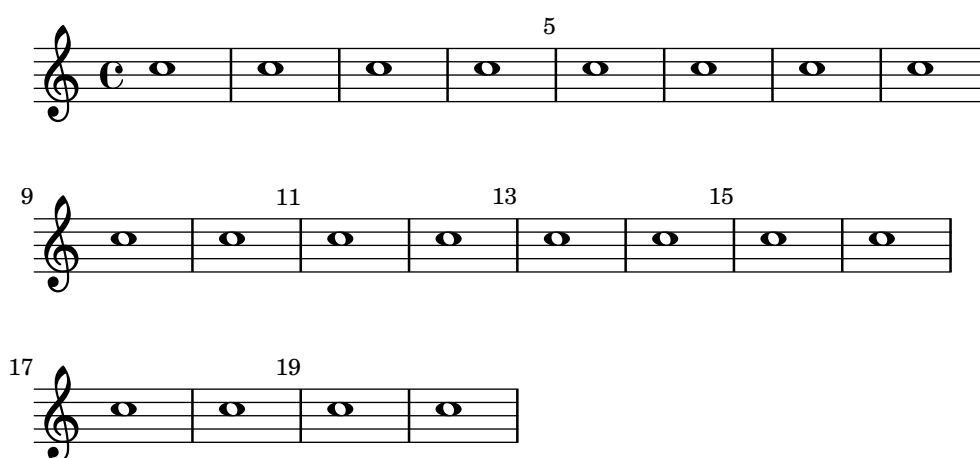
```
\relative c' {
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \set Score.currentBarNumber = #11
  % Permit first bar number to be printed
  \bar ""
  % Print a bar number every second measure
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
  c1 | c | c | c | c
  \break
  c1 | c | c | c | c
}
```



Printing bar numbers with changing regular intervals

The bar number interval can be changed by changing the context function `{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
}
```

*Printing bar numbers inside boxes or circles*

Bar numbers can also be printed inside boxes or circles.

```
\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 "|."
}
```



*Alternative bar numbering*

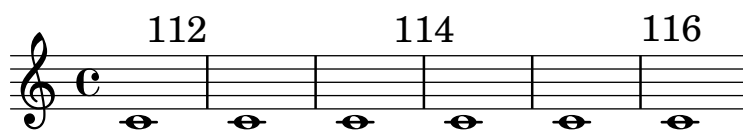
Two alternative methods for bar numbering can be set, especially for when using repeated music.

```
\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
}
```

Aligning bar numbers

Bar numbers by default are right-aligned to their parent object. This is usually the left edge of a line or, if numbers are printed within a line, the left hand side of a bar line. The numbers may also be positioned directly over the bar line or left-aligned to the bar line.

```
\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
}
```

*Removing bar numbers from a score*

Bar numbers can be removed entirely by removing the `Bar_number_engraver` from the `Score` context.

```
\layout {
  \context {
    \Score
    \omit BarNumber
    % or:
    %\remove "Bar_number_engraver"
  }
}

\relative c'' {
  c4 c c c \break
  c4 c c c
}
```



Vegeu també

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “BarNumber” in *Referència de funcionament intern*, Secció “Bar_number_engraver” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Bar numbers may collide with the top of the `StaffGroup` bracket, if there is one. To solve this, the `padding` property of `BarNumber` can be used to position the number correctly. See Secció “StaffGroup” in *Referència de funcionament intern* and Secció “BarNumber” in *Referència de funcionament intern* for more.

Bar and bar number checks

Bar checks help detect errors in the entered durations. A bar check may be entered using the bar symbol, |, at any place where a bar line is expected to fall. If bar check lines are encountered at other places, a list of warnings is printed in the log file, showing the line numbers and lines in which the bar checks failed. In the next example, the second bar check will signal an error.

```
\time 3/4 c2 e4 | g2 |
```

An incorrect duration can result in a completely garbled score, especially if the score is polyphonic, so a good place to start correcting input is by scanning for failed bar checks and incorrect durations.

If successive bar checks are off by the same musical interval, only the first warning message is displayed. This allows the warning to focus on the source of the timing error.

Bar checks can also be inserted in lyrics:

```
\lyricmode {
  \time 2/4
  Twin -- kle | Twin -- kle |
}
```

Note that bar check marks in lyrics are evaluated at the musical moment when the syllable *following* the check mark is processed. If the lyrics are associated with the notes of a voice which has a rest at the beginning of a bar, then no syllable can be located at the start of that bar and a warning will be issued if a bar check mark is placed in the lyrics at that position.

It is also possible to redefine the action taken when a bar check or pipe symbol, |, is encountered in the input, so that it does something other than a bar check. This is done by assigning a music expression to "|". In the following example | is set to insert a double bar line wherever it appears in the input, rather than checking for end of bar.

```
"|" = \bar "||"
{
  c'2 c' |
  c'2 c'
  c'2 | c'
  c'2 c'
}
```



When copying large pieces of music, it can be helpful to check that the LilyPond bar number corresponds to the original that you are entering from. This can be checked with `\barNumberCheck`, for example,

```
\barNumberCheck #123
```

will print a warning if the `currentBarNumber` is not 123 when it is processed.

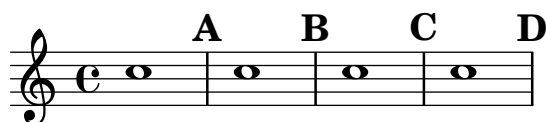
Vegeu també

Snippets: Secció “Rhythms” in *Fragments de codi*.

Rehearsal marks

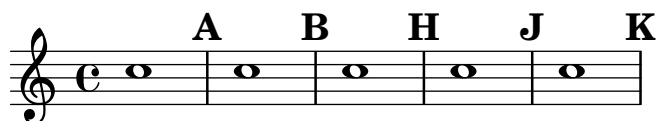
To print a rehearsal mark, use the `\mark` command.

```
\relative c'' {
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
}
```



The mark is incremented automatically if you use `\mark \default`, but you can also use an integer argument to set the mark manually. The value to use is stored in the property `rehearsalMark`.

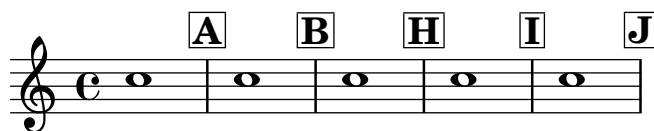
```
\relative c'' {
  c1 \mark \default
  c1 \mark \default
  c1 \mark #8
  c1 \mark \default
  c1 \mark \default
}
```



The letter ‘I’ is skipped in accordance with engraving traditions. If you wish to include the letter ‘I’, then use one of the following commands, depending on which style of rehearsal mark you want (letters only, letters in a hollow box, or letters in a hollow circle).

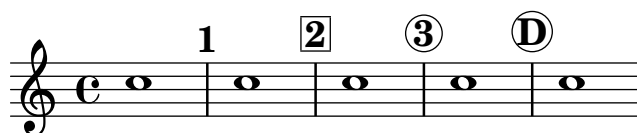
```
\set Score.markFormatter = #format-mark-alphabet
\set Score.markFormatter = #format-mark-box-alphabet
\set Score.markFormatter = #format-mark-circle-alphabet

\relative c'' {
  \set Score.markFormatter = #format-mark-box-alphabet
  c1 \mark \default
  c1 \mark \default
  c1 \mark #8
  c1 \mark \default
  c1 \mark \default
}
```

The style is defined by the property `markFormatter`. It is a function taking the current mark (an integer) and the current context as argument. It should return a markup object. In the following example, `markFormatter` is set to a pre-defined procedure. After a few measures, it is set to a procedure that produces a boxed number.

```
\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
}
```



The file `scm/translation-functions.scm` contains the definitions of `format-mark-numbers` (the default format), `format-mark-box-numbers`, `format-mark-letters` and `format-mark-box-letters`. These can be used as inspiration for other formatting functions.

You may use `format-mark-barnumbers`, `format-mark-box-barnumbers`, and `format-mark-circle-barnumbers` to get bar numbers instead of incremented numbers or letters.

Other styles of rehearsal mark can be specified manually:

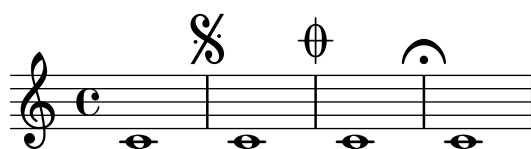
```
\mark "A1"
```

Note that `Score.markFormatter` does not affect marks specified in this manner. However, it is possible to apply a `\markup` to the string.

```
\mark \markup{ \box A1 }
```

Music glyphs (such as the segno sign) may be printed inside a `\mark`

```
\relative c' {
  c1 \mark \markup { \musicglyph #"scripts.segno" }
  c1 \mark \markup { \musicglyph #"scripts.coda" }
  c1 \mark \markup { \musicglyph #"scripts.ufermata" }
  c1
}
```



See Secció A.8 [The Feta font], pàgina 651, for a list of symbols which may be printed with `\musicglyph`.

For common tweaks to the positioning of rehearsal marks, see Secció 1.8.2 [Formatting text], pàgina 237. For more precise control, see `break-alignable-interface` in Secció 5.5.1 [Aligning objects], pàgina 618.

The file `scm/translation-functions.scm` contains the definitions of `format-mark-numbers` and `format-mark-letters`. They can be used as inspiration for other formatting functions.

Vegeu també

Notation Reference: Secció A.8 [The Feta font], pàgina 651, Secció 1.8.2 [Formatting text], pàgina 237, Secció 5.5.1 [Aligning objects], pàgina 618.

Installed Files: `scm/translation-functions.scm`.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “MarkEvent” in *Referència de funcionament intern*, Secció “Mark_engraver” in *Referència de funcionament intern*, Secció “RehearsalMark” in *Referència de funcionament intern*.

1.2.6 Special rhythmic concerns

Grace notes

Grace notes are musical ornaments, printed in a smaller font, that take up no additional logical time in a measure.

```
\relative {
  c' '4 \grace b16 a4(
    \grace { b16 c16 } a2)
}
```



There are three other types of grace notes possible; the *acciaccatura* – an unmeasured grace note indicated by a slurred note with a slashed stem – and the *appoggiatura*, which takes a fixed fraction of the main note it is attached to and prints without the slash. It is also possible to write a grace note with a slashed stem, like the *acciaccatura* but without the slur, so as to place it between notes that are slurred themselves, using the `\slashedGrace` function.

```
\relative {
  \acciaccatura d' '8 c4
  \appoggiatura e8 d4
  \acciaccatura { g16 f } e2
  \slashedGrace a,8 g4
  \slashedGrace b16 a4(
  \slashedGrace b8 a2)
}
```



The placement of grace notes is synchronized between different staves. In the following example, there are two sixteenth grace notes for every eighth grace note

<<

```

\new Staff \relative { e''2 \grace { c16 d e f } e2 }
\new Staff \relative { c''2 \grace { g8 b } c2 }
>>

```

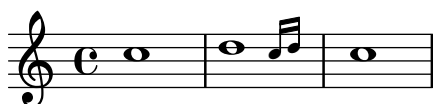


If you want to end a note with a grace, use the `\afterGrace` command. It takes two arguments: the main note, and the grace notes following the main note.

```

\relative { c''1 \afterGrace d1 { c16[ d] } c1 }

```



This will place the grace notes *after* the start of the main note. The point of time where the grace notes are placed is a given fraction of the main note's duration. The default setting of

```

afterGraceFraction = 3/4

```

may be redefined at top level. Individual `\afterGrace` commands may have the fraction specified right after the command itself instead.

The following example shows the results from setting with the default space, setting it at 15/16, and finally at 1/2 of the main note.

```

<<
\new Staff \relative {
  c''1 \afterGrace d1 { c16[ d] } c1
}
\new Staff \relative {
  c''1 \afterGrace 15/16 d1 { c16[ d] } c1
}
\new Staff \relative {
  c''1 \afterGrace 1/2 d1 { c16[ d] } c1
}
>>

```



The effect of `\afterGrace` can also be achieved using spacers. The following example places the grace note after a space lasting 7/8 of the main note.

```

\new Voice \relative {

```

```

<<
  { d''1^\trill_( }
  { s2 s4. \grace { c16 d } }
>>
c1)
}

```



A `\grace` music expression will introduce special typesetting settings, for example, to produce smaller type, and set directions. Hence, when introducing layout tweaks to override the special settings, they should be placed inside the grace expression. The overrides should also be reverted inside the grace expression. Here, the grace note's default stem direction is overridden and then reverted.

```

\new Voice \relative {
  \acciaccatura {
    \stemDown
    f''16->
    \stemNeutral
  }
  g4 e c2
}

```



Fragments de codi seleccionats

Using grace note slashes with normal heads

The slash through the stem found in acciaccaturas can be applied in other situations.

```

\relative c'' {
  \override Flag.stroke-style = #"grace"
  c8( d2) e8( f4)
}

```



Tweaking grace layout within music

The layout of grace expressions can be changed throughout the music using the functions `add-grace-property` and `remove-grace-property`. The following example undefines the `Stem` direction for this grace, so that stems do not always point up, and changes the default note heads to crosses.

```

\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
}
}
}

```



Redefining grace note global defaults

The global defaults for grace notes are stored in the identifiers `startGraceMusic`, `stopGraceMusic`, `startAcciaccaturaMusic`, `stopAcciaccaturaMusic`, `startAppoggiaturaMusic` and `stopAppoggiaturaMusic`, which are defined in the file `ly/grace-init.ly`. By redefining them other effects may be obtained.

```

startAcciaccaturaMusic = {
  <>(
    \override Flag.stroke-style = #"grace"
    \slurDashed
  )
}

stopAcciaccaturaMusic = {
  \revert Flag.stroke-style
  \slurSolid
  <>
}

\relative c'' {
  \acciaccatura d8 c1
}

```



Positioning grace notes with floating space

Setting the property '`strict-grace-spacing`' makes the musical columns for grace notes 'floating', i.e., decoupled from the non-grace notes: first the normal notes are spaced, then the (musical columns of the) graces are put left of the musical columns for the main notes.

```

\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
    }
  }
}

```



Vegeu també

Music Glossary: Secció “grace notes” in *Glossari musical*, Secció “acciaccatura” in *Glossari musical*, Secció “appoggiatura” in *Glossari musical*.

Notation Reference: [Scaling durations], pàgina 52, [Manual beams], pàgina 93.

Installed Files: `ly/grace-init.ly`.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “GraceMusic” in *Referència de funcionament intern*, Secció “Grace_beam_engraver” in *Referència de funcionament intern*, Secció “Grace_auto_beam_engraver” in *Referència de funcionament intern*, Secció “Grace_engraver” in *Referència de funcionament intern*, Secció “Grace_spacing_engraver” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

A multi-note beamed *acciaccatura* is printed without a slash, and looks exactly the same as a multi-note beamed *appoggiatura*.

Grace note synchronization can also lead to surprises. Staff notation, such as key signatures, bar lines, etc., are also synchronized. Take care when you mix staves with grace notes and staves without, for example,

```
<<
  \new Staff \relative { e''4 \bar ".|:" \grace c16 d2. }
  \new Staff \relative { c''4 \bar ".|:" d2. }
>>
```



This can be remedied by inserting grace skips of the corresponding durations in the other staves. For the above example

```
<<
  \new Staff \relative { e''4 \bar ".|:" \grace c16 d2. }
  \new Staff \relative { c''4 \bar ".|:" \grace s16 d2. }
>>
```



Please make sure that you use the `\grace` command for the spacer part, even if the visual part uses `\acciaccatura` or `\appoggiatura` because otherwise an ugly slur fragment will be printed, connecting the invisible grace note with the following note.

Grace sections should only be used within sequential music expressions. Nesting or juxtaposing grace sections is not supported, and might produce crashes or other errors.

Each grace note in MIDI output has a length of 1/4 of its actual duration. If the combined length of the grace notes is greater than the length of the preceding note a “Going back in MIDI time” error will be generated. Either make the grace notes shorter in duration, for example:

```
c'8 \acciaccatura { c'8[ d' e' f' g'] }
```

becomes:

```
c'8 \acciaccatura { c'16[ d' e' f' g'] }
```

Or explicitly change the musical duration:

```
c'8 \acciaccatura { \scaleDurations 1/2 { c'8[ d' e' f' g'] } }
```

See [Scaling durations], pàgina 52.

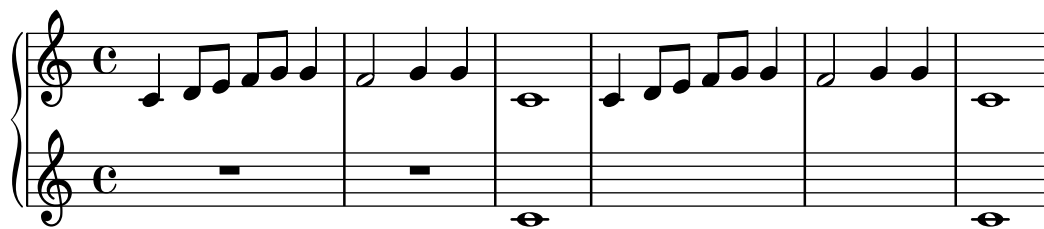
Aligning to cadenzas

In an orchestral context, cadenzas present a special problem: when constructing a score that includes a measured cadenza or other solo passage, all other instruments should skip just as many notes as the length of the cadenza, otherwise they will start too soon or too late.

One solution to this problem is to use the functions `mmrest-of-length` and `skip-of-length`. These Scheme functions take a defined piece of music as an argument and generate a multi-measure rest or `\skip` exactly as long as the piece.

```
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
  }
>>
```



Vegeu també

Music Glossary: Secció “cadenza” in *Glossari musical*.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Time administration

Time is administered by the `Timing_translator`, which by default is to be found in the `Score` context. An alias, `Timing`, is added to the context in which the `Timing_translator` is placed. To ensure that the `Timing` alias is available, you may need to explicitly instantiate the containing context (such as `Voice` or `Staff`).

The following properties of `Timing` are used to keep track of timing within the score.

`currentBarNumber`

The current measure number. For an example showing the use of this property see [Bar numbers], pàgina 104.

`measureLength`

The length of the measures in the current time signature. For a 4/4 time this is 1, and for 6/8 it is 3/4. Its value determines when bar lines are inserted and how automatic beams should be generated.

`measurePosition`

The point within the measure where we currently are. This quantity is reset by subtracting `measureLength` whenever `measureLength` is reached or exceeded. When that happens, `currentBarNumber` is incremented.

`timing` If set to true, the above variables are updated for every time step. When set to false, the engraver stays in the current measure indefinitely.

Timing can be changed by setting any of these variables explicitly. In the next example, the default 4/4 time signature is printed, but `measureLength` is set to 5/4. At 4/8 through the third measure, the `measurePosition` is advanced by 1/8 to 5/8, shortening that bar by 1/8. The next bar line then falls at 9/8 rather than 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 |
}
```



As the example illustrates, `ly:make-moment n/m` constructs a duration of n/m of a whole note. For example, `ly:make-moment 1/8` is an eighth note duration and `ly:make-moment 7/16` is the duration of seven sixteenths notes.

Vegeu també

Notation Reference: [Bar numbers], pàgina 104, [Unmetered music], pàgina 73.

Snippets: Secció “Rhythms” in *Fragments de codi*.

Internals Reference: Secció “Timing_translator” in *Referència de funcionament intern*, Secció “Score” in *Referència de funcionament intern*.

1.3 Expressive marks

RONDO
Allegro

This section lists various expressive marks that can be created in a score.

1.3.1 Expressive marks attached to notes

This section explains how to create expressive marks that are attached to notes: articulations, ornamentations, and dynamics. Methods to create new dynamic markings are also discussed.

Articulations and ornamentations

A variety of symbols that denote articulations, ornamentations, and other performance indications can be attached to a note using this syntax:

`note\name`

The possible values for `name` are listed in Secció A.14 [List of articulations], pàgina 732. For example:

```
\relative {
  c'4\staccato c\mordent b2\turn
  c1\fermata
}
```



Some of these articulations have shorthands for easier entry. Shorthands are appended to the note name, and their syntax consists of a dash – followed by a symbol signifying the articulation. Predefined shorthands exist for *marcato*, *stopped*, *tenuto*, *staccatissimo*, *accent*, *staccato*, and *portato*. Their corresponding output appears as follows:

```
\relative {
  c'4-^ c-+ c-- c-!
  c4-> c-. c2-_
}
```



The rules for the default placement of articulations are defined in `scm/script.scm`. Articulations and ornamentations may be manually placed above or below the staff; see Secció 5.4.2 [Direction and placement], pàgina 602.

Articulations are `Script` objects. Their properties are described more fully in Secció “Script” in *Referència de funcionament intern*.

Articulations can be attached to rests as well as notes but they cannot be attached to multi-measure rests. A special predefined command, `\fermataMarkup`, is available for attaching a fermata to a multi-measure rest (and only a multi-measure rest). This creates a `MultiMeasureRestText` object.

```
\override Script.color = #red
\override MultiMeasureRestText.color = #blue
a'2\fermata r\fermata
R1\fermataMarkup
```



In addition to articulations, text and markups can be attached to notes. See [Text scripts], pàgina 230.

For more information about the ordering of `Scripts` and `TextScripts` that are attached to the notes, see Secció “Placement of objects” in *Manual d’aprenentatge*.

Fragments de codi seleccionats

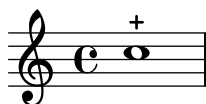
Modifying default values for articulation shorthand notation

The shorthands are defined in ‘`ly/script-init.ly`’, where the variables `dashHat`, `dashPlus`, `dashDash`, `dashBar`, `dashLarger`, `dashDot`, and `dashUnderscore` are assigned default values. The default values for the shorthands can be modified. For example, to associate the `++` (`dashPlus`) shorthand with the trill symbol instead of the default `+` symbol, assign the value `trill` to the variable `dashPlus`:

```
\relative c'' { c1-+ }

dashPlus = "trill"

\relative c'' { c1-+ }
```



Controlling the vertical ordering of scripts

The vertical ordering of scripts is controlled with the '`script-priority`' property. The lower this number, the closer it will be put to the note. In this example, the `TextScript` (the sharp symbol) first has the lowest priority, so it is put lowest in the first example. In the second, the `prall trill` (the `Script`) has the lowest, so it is on the inside. When two objects have the same priority, the order in which they are entered determines which one comes first.

```
\relative c'' {
  \once \override TextScript.script-priority = #-100
  a2^\prall^\markup { \sharp }

  \once \override Script.script-priority = #-100
  a2^\prall^\markup { \sharp }
}
```



Creating a delayed turn

Creating a delayed turn, where the lower note of the turn uses the accidental, requires several overrides. The `outside-staff-priority` property must be set to `#f`, as otherwise this would take precedence over the `avoid-slur` property. Changing the fractions `2/3` and `1/3` adjusts the horizontal position.

```
\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
    }
  >>
```

}



Vegeu també

Music Glossary: Secció “tenuto” in *Glossari musical*, Secció “accent” in *Glossari musical*, Secció “staccato” in *Glossari musical*, Secció “portato” in *Glossari musical*.

Learning Manual: Secció “Placement of objects” in *Manual d’aprenentatge*.

Notation Reference: [Text scripts], pàgina 230, Secció 5.4.2 [Direction and placement], pàgina 602, Secció A.14 [List of articulations], pàgina 732, [Trills], pàgina 145.

Installed Files: `scm/script.scm`.

Snippets: Secció “Expressive marks” in *Fragments de codi*.

Internals Reference: Secció “Script” in *Referència de funcionament intern*, Secció “TextScript” in *Referència de funcionament intern*.

Dynamics

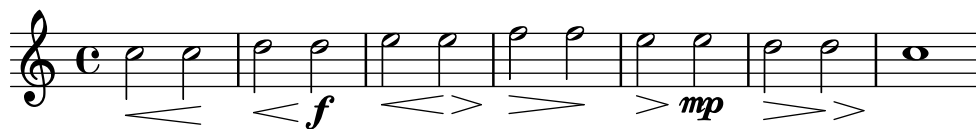
Absolute dynamic marks are specified using a command after a note, such as `c4\ff`. The available dynamic marks are `\ppppp`, `\pppp`, `\ppp`, `\pp`, `\p`, `\mp`, `\mf`, `\f`, `\ff`, `\fff`, `\ffff`, `\fffff`, `\fp`, `\sf`, `\sff`, `\sp`, `\spp`, `\sfz`, and `\rfz`. Dynamic marks may be manually placed above or below the staff; see Secció 5.4.2 [Direction and placement], pàgina 602.

```
\relative c'' {
  c2\ppp c\mp
  c2\rfz c^\mf
  c2_\spp c^\ff
}
```



A *crescendo* mark is started with `\<` and terminated with `\!`, an absolute dynamic, or an additional crescendo or decrescendo mark. A *decrescendo* mark is started with `\>` and is also terminated with `\!`, an absolute dynamic, or another crescendo or decrescendo mark. `\cr` and `\decr` may be used instead of `\<` and `\>`. *Hairpins* are engraved by default using this notation.

```
\relative c'' {
  c2\< c\!
  d2\< d\f
  e2\< e\>
  f2\> f\!
  e2\> e\mp
  d2\> d\>
  c1\!
}
```



A hairpin that is terminated with `\!` will end at the right edge of the note that has the `\!` assigned to it. In the case where it is terminated with the start of another *crescendo* or *decrescendo* mark, it will end at the centre of the note that has the next `\<` or `\>` assigned to it. The next hairpin will then start at the right edge of the same note instead of the usual left edge had it been terminated with `\!` before. A hairpin ending on a downbeat will stop at the preceding bar line.

```
\relative {
  c''1\< | c4 a c\< a | c4 a c\! a\< | c4 a c a\!
}
```



Hairpins that are terminated with absolute dynamic marks instead of `\!` will also be engraved in a similar way. However, the length of the absolute dynamic itself can alter where the preceding hairpin ends.

```
\relative {
  c''1\< | c4 a c\mf a | c1\< | c4 a c\ffff a
}
```



Spacer rests are needed to engrave multiple marks on one note. This is particularly useful when adding a *crescendo* and *decrescendo* to the same note:

```
\relative {
  c''4\< c\! d\> e\!
  << f1 { s4 s4\< s4\> s4\! } >>
}
```



The `\espressivo` command can be used to indicate a crescendo and decrescendo on the same note. However, be warned that this is implemented as an articulation, not a dynamic.

```
\relative {
  c''2 b4 a
  g1\espressivo
}
```



Textual crescendo marks begin with `\cresc.` Textual decrescendos begin with `\decrec` or `\dim.` Extender lines are engraved as required.

```
\relative {
  g'8\cresc a b c b c d e\mf |
  f8\decrec e d c e\> d c b |
  a1\dim ~ |
  a2. r4\! |
}
```



Textual marks for dynamic changes can also replace hairpins:

```
\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\! |
}
```



To create new absolute dynamic marks or text that should be aligned with dynamics, see [New dynamic marks], pàgina 128.

Vertical positioning of dynamics is handled by Secció “DynamicLineSpanner” in *Referència de funcionament intern*.

A **Dynamics** context is available to engrave dynamics on their own horizontal line. Use spacer rests to indicate timing. (Notes in a **Dynamics** context will also take up musical time, but will not be engraved.) The **Dynamics** context can usefully contain some other items such as text scripts, text spanners, and piano pedal marks.

```
<<
\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 |
}
>>

```



Instruccions predefinides

```

\dynamicUp, \dynamicDown, \dynamicNeutral, \crescTextCresc, \dimTextDim,
\dimTextDecr, \dimTextDecresc, \crescHairpin, \dimHairpin.

```

Fragments de codi seleccionats

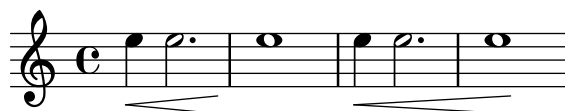
Setting hairpin behavior at bar lines

If the note which ends a hairpin falls on a downbeat, the hairpin stops at the bar line immediately preceding. This behavior can be controlled by overriding the 'to-barline property.

```

\relative c'' {
  e4\< e2.
  e1\!
  \override Hairpin.to-barline = ##f
  e4\< e2.
  e1\!
}

```



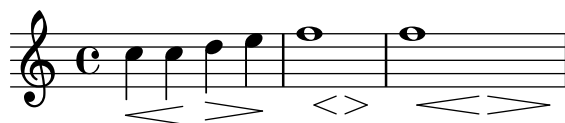
Setting the minimum length of hairpins

If hairpins are too short, they can be lengthened by modifying the minimum-length property of the Hairpin object.

```

\relative c'' {
  c4\< c\! d\> e\!
  << f1 { s4 s\< s\> s\! } >>
  \override Hairpin.minimum-length = #5
  << f1 { s4 s\< s\> s\! } >>
}

```



Printing hairpins using al niente notation

Hairpin dynamics may be printed with a circled tip ("al niente" notation) by setting the circled-tip property of the Hairpin object to #t.

```

\relative c'' {
  \override Hairpin.circled-tip = ##t

```

```

c2\< c\!
c4\> c\< c2\!
}

```



Printing hairpins in various styles

Hairpin dynamics may be created in a variety of styles.

```

\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\!
}

```



Vertically aligned dynamics and textscripts

All `DynamicLineSpanner` objects (hairpins and dynamic texts) are placed with their reference line at least 'staff-padding' from the staff, unless other notation forces them to be farther. Setting 'staff-padding' to a sufficiently large value aligns the dynamics.

The same idea, together with `\textLengthOn`, is used to align the text scripts along their baseline.

```

music = \relative c' {
  a'2\p b\f
  e4\p f\f\> g, b\p
  c2^\markup { \huge gorgeous } c^\markup { \huge fantastic }
}

```



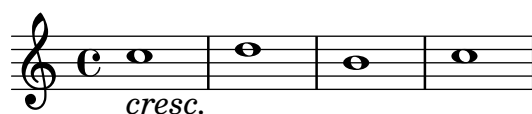
```
{
  \music
  \break
  \override DynamicLineSpanner.staff-padding = #3
  \textLengthOn
  \override TextScript.staff-padding = #1
  \music
}
```



Hiding the extender line for text dynamics

Text style dynamic changes (such as *cresc.* and *dim.*) are printed with a dashed line showing their extent. This line can be suppressed in the following way:

```
\relative c'' {
  \override DynamicTextSpanner.style = #'none
  \crescTextCresc
  c1\< | d | b | c\!
}
```



Changing text and spanner styles for text dynamics

The text used for crescendos and decrescendos can be changed by modifying the context properties `crescendoText` and `decrescendoText`.

The style of the spanner line can be changed by modifying the 'style' property of `DynamicTextSpanner`. The default value is 'dashed-line', and other possible values include 'line', 'dotted-line' and '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
}
```



Vegeu també

Music Glossary: Secció “al niente” in *Glossari musical*, Secció “crescendo” in *Glossari musical*, Secció “decrescendo” in *Glossari musical*, Secció “hairpin” in *Glossari musical*.

Learning Manual: Secció “Articulation and dynamics” in *Manual d’aprenentatge*.

Notation Reference: Secció 5.4.2 [Direction and placement], pàgina 602, [New dynamic marks], pàgina 128, Secció 3.5.9 [Enhancing MIDI output], pàgina 518, Secció 3.5.4 [Controlling MIDI dynamics], pàgina 509.

Snippets: Secció “Expressive marks” in *Fragments de codi*.

Internals Reference: Secció “DynamicText” in *Referència de funcionament intern*, Secció “Hairpin” in *Referència de funcionament intern*, Secció “DynamicLineSpanner” in *Referència de funcionament intern*, Secció “Dynamics” in *Referència de funcionament intern*.

New dynamic marks

The easiest way to create dynamic indications is to use `\markup` objects.

```
moltoF = \markup { molto \dynamic f }
```

```
\relative {
  <d' e>16_\moltoF <d e>
  <d e>2..
}
```



In markup mode, editorial dynamics (within parentheses or square brackets) can be created. The syntax for markup mode is described in Secció 1.8.2 [Formatting text], pàgina 237.

```
roundF = \markup {
  \center-align \concat { \bold { \italic ( }
    \dynamic f \bold { \italic ) } } }
boxF = \markup { \bracket { \dynamic f } }
\relative {
  c'1_\roundF
  c1_\boxF
}
```



Simple, centered dynamic marks are easily created with the `make-dynamic-script` function.

```
sfzp = #(make-dynamic-script "sfzp")
\relative {
  c'4 c c\sfpz c
}
```



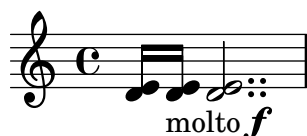
In general, `make-dynamic-script` takes any markup object as its argument. The dynamic font only contains the characters `f`, `m`, `p`, `r`, `s` and `z`, so if a dynamic mark that includes plain text or punctuation symbols is desired, markup commands that reverts font family and font encoding to normal text should be used, for example `\normal-text`. The interest of using `make-dynamic-script` instead of an ordinary markup is ensuring the vertical alignment of markup objects and hairpins that are attached to the same note head.

```
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
}
```



The Scheme form of markup mode may be used instead. Its syntax is explained in Secció “Markup construction in Scheme” in *Extendre*.

```
moltoF = #(make-dynamic-script
  (markup #:normal-text "molto"
    #:dynamic "f"))
\relative {
  <d' e>16 <d e>
  <d e>2..\moltoF
}
```



To left-align the dynamic text rather than centering it on a note use a `\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
}

```



Font settings in markup mode are described in [Selecting font and font size], pàgina 239.

Vegeu també

Notation Reference: Secció 1.8.2 [Formatting text], pàgina 237, [Selecting font and font size], pàgina 239, Secció 3.5.9 [Enhancing MIDI output], pàgina 518, Secció 3.5.4 [Controlling MIDI dynamics], pàgina 509.

Extending LilyPond: Secció “Markup construction in Scheme” in *Extendre*.

Snippets: Secció “Expressive marks” in *Fragments de codi*.

1.3.2 Expressive marks as curves

This section explains how to create various expressive marks that are curved: normal slurs, phrasing slurs, breath marks, falls, and doits.

Slurs

Slurs are entered using parentheses:

Nota: In polyphonic music, a slur must be terminated in the same voice it began.

```
\relative {
  f''4( g a) a8 b(
  a4 g2 f4)
  <c e>2( <b d>2)
}
```



Slurs may be manually placed above or below the staff; see Secció 5.4.2 [Direction and placement], pàgina 602.

Simultaneous or overlapping slurs require special attention. Most occurrences of outer slurs actually indicate phrasing, and phrasing slurs may overlap a regular slur, see [Phrasing slurs], pàgina 133. When multiple regular slurs are needed in a single **Voice**, matching slur starts and ends need to be labelled by preceding them with \= followed by an identifying key (a symbol or non-negative integer).

$$\begin{aligned} & \text{\fixed c' } \{ \\ & \quad \langle c \sim f \rangle = 1 (\text{\g} = 2 (\text{\> } \langle c \text{\e} \rangle = 1) \text{\a} \rangle = 2) \text{\>} \\ & \} \end{aligned}$$



Slurs can be solid, dotted, or dashed. Solid is the default slur style:

```
\relative {
  c'4( e g2)
  \slurDashed
  g4( e c2)
  \slurDotted
  c4( e g2)
  \slurSolid
  g4( e c2)
}
```



Slurs can also be made half-dashed (the first half dashed, the second half solid) or half-solid (the first half solid, the second half dashed):

```
\relative {
  c'4( e g2)
  \slurHalfDashed
  g4( e c2)
  \slurHalfSolid
  c4( e g2)
  \slurSolid
  g4( e c2)
}
```



Custom dash patterns for slurs can be defined:

```
\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)
}
```



Instruccions predefinides

`\slurUp`, `\slurDown`, `\slurNeutral`, `\slurDashed`, `\slurDotted`, `\slurHalfDashed`,
`\slurHalfSolid`, `\slurDashPattern`, `\slurSolid`.

Using double slurs for legato chords

```
\relative c' {
  \set doubleSlurs = ##t
  <c e>4( <d f> <c e> <d f>)
}
```



```
\relative c'' {
  \override TextScript.avoid-slur = #'inside
  \override TextScript.outside-staff-priority = ##f
  c2(^\markup { \halign #-10 \natural } d4.) c8
}
```



```
\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)  
}
```



Vegeu també

Music Glossary: Secció “slur” in *Glossari musical*.

Learning Manual: Secció “On the un-nestedness of brackets and ties” in *Manual d’aprenentatge*.

Notation Reference: Secció 5.4.2 [Direction and placement], pàgina 602, [Phrasing slurs], pàgina 133.

Snippets: Secció “Expressive marks” in *Fragments de codi*.

Internals Reference: Secció “Slur” in *Referència de funcionament intern*.

Phrasing slurs

Phrasing slurs (or phrasing marks) that indicate a musical sentence are written using the commands `\(` and `\)` respectively:

```
\relative {
  c'4\ ( d( e) f(
  e2) d\ )
}
```



Typographically, a phrasing slur behaves almost exactly like a normal slur. However, they are treated as different objects; a `\slurUp` will have no effect on a phrasing slur. Phrasing may be manually placed above or below the staff; see Secció 5.4.2 [Direction and placement], pàgina 602.

Simultaneous or overlapping phrasing slurs are entered using `\=` as with regular slurs, see [Slurs], pàgina 130.

Phrasing slurs can be solid, dotted, or dashed. Solid is the default style for phrasing slurs:

```
\relative {
  c'4\ ( e g2\ )
  \phrasingSlurDashed
  g4\ ( e c2\ )
  \phrasingSlurDotted
  c4\ ( e g2\ )
  \phrasingSlurSolid
  g4\ ( e c2\ )
}
```



Phrasing slurs can also be made half-dashed (the first half dashed, the second half solid) or half-solid (the first half solid, the second half dashed):

```
\relative {
  c'4\ ( e g2\ )
  \phrasingSlurHalfDashed
  g4\ ( e c2\ )
}
```

```

\phrasingSlurHalfSolid
c4\ ( e g2\ )
\phrasingSlurSolid
g4\ ( e c2\ )
}

```



Custom dash patterns for phrasing slurs can be defined:

```

\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\ )
}

```



Dash pattern definitions for phrasing slurs have the same structure as dash pattern definitions for slurs. For more information about complex dash patterns, see the snippets under [Slurs], pàgina 130.

Instruccions predefinides

```

\phrasingSlurUp, \phrasingSlurDown, \phrasingSlurNeutral, \phrasingSlurDashed,
\phrasingSlurDotted, \phrasingSlurHalfDashed, \phrasingSlurHalfSolid,
\phrasingSlurDashPattern, \phrasingSlurSolid.

```

Vegeu també

Learning Manual: Secció “On the un-nestedness of brackets and ties” in *Manual d’aprenentatge*.

Notation Reference: Secció 5.4.2 [Direction and placement], pàgina 602, [Slurs], pàgina 130.

Snippets: Secció “Expressive marks” in *Fragments de codi*.

Internals Reference: Secció “PhrasingSlur” in *Referència de funcionament intern*.

Breath marks

Breath marks are entered using `\breathe`:

```
{ c'2. \breathe d'4 }
```



Unlike other expressive marks, a breath mark is not associated with the preceding note but is a separate music event. So all the expressive marks which are attached to the preceding note,

any square brackets indicating manual beams, and any brackets indicating slurs and phrasing slurs must be placed before `\breathe`.

A breath mark will end an automatic beam; to override this behavior, see [Manual beams], pàgina 93.

```
\relative { c''8 \breathe d e f g2 }
```



Musical indicators for breath marks in ancient notation, *divisiones*, are supported. For details, see [Divisiones], pàgina 437.

Fragments de codi seleccionats

Changing the breath mark symbol

The glyph of the breath mark can be tuned by overriding the `text` property of the `BreathingSign` layout object with any markup text.

```
\relative c'' {
  c2
  \override BreathingSign.text =
    \markup { \musicglyph #"scripts.rvarcomma" }
  \breathe
  d2
}
```



Using a tick as the breath mark symbol

Vocal and wind music frequently uses a tick mark as a breathing sign. This indicates a breath that subtracts a little time from the previous note rather than causing a short pause, which is indicated by the comma breath mark. The mark can be moved up a little to take it away from the stave.

```
\relative c'' {
  c2
  \breathe
  d2
  \override BreathingSign.Y-offset = #2.6
  \override BreathingSign.text =
    \markup { \musicglyph #"scripts.tickmark" }
  c2
  \breathe
  d2
}
```



Inserting a caesura

Caesura marks can be created by overriding the 'text' property of the `BreathingSign` object. A curved caesura mark is also available.

```
\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
}
```



Vegeu també

Music Glossary: Secció “caesura” in *Glossari musical*.

Notation Reference: [Divisiones], pàgina 437.

Snippets: Secció “Expressive marks” in *Fragments de codi*.

Internals Reference: Secció “BreathingEvent” in *Referència de funcionament intern*, Secció “BreathingSign” in *Referència de funcionament intern*, Secció “Breathing-sign-engraver” in *Referència de funcionament intern*.

Falls and doits

Falls and *doits* can be added to notes using the `\bendAfter` command. The direction of the fall or doit is indicated with a plus or minus (up or down). The number indicates the pitch interval that the fall or doit will extend *beyond* the main note.

```
\relative c'' {
  c2\bendAfter #+4
  c2\bendAfter #-4
  c2\bendAfter #+6.5
  c2\bendAfter #-6.5
  c2\bendAfter #+8
  c2\bendAfter #-8
}
```



Fragments de codi seleccionats

Adjusting the shape of falls and doits

The `shortest-duration-space` property may be tweaked to adjust the shape of falls and doits.

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

```

\override Score.SpacingSpanner.shortest-duration-space = #4.0
c2-\bendAfter #5
c2-\bendAfter #-4.75
c2-\bendAfter #8.5
c2-\bendAfter #-6
}

```



Vegeu també

Music Glossary: Secció “fall” in *Glossari musical*, Secció “doit” in *Glossari musical*.

Snippets: Secció “Expressive marks” in *Fragments de codi*.

1.3.3 Expressive marks as lines

This section explains how to create various expressive marks that follow a linear path: glissandos, arpeggios, and trills.

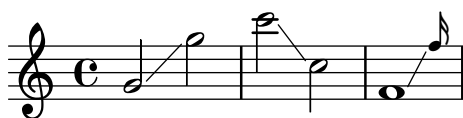
Glissando

A *glissando* is created by appending `\glissando` to a note:

```

\relative {
  g'2\glissando g'
  c2\glissando c,
  \afterGrace f,1\glissando f'16
}

```

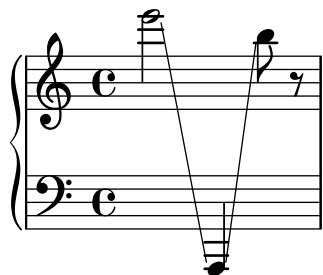


A glissando can connect notes across staves:

```

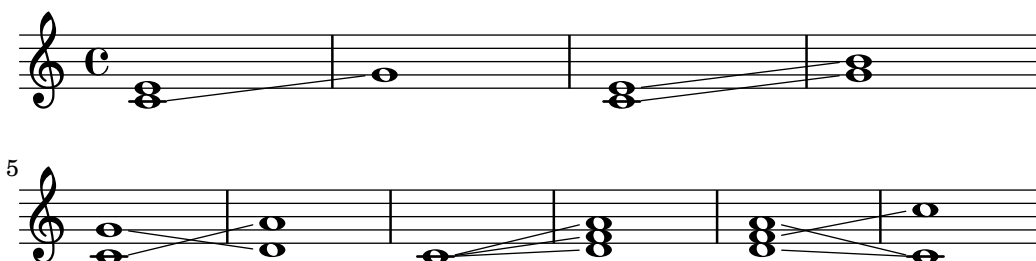
\new PianoStaff <<
  \new Staff = "right" {
    e''2\glissando
    \change Staff = "left"
    a,,4\glissando
    \change Staff = "right"
    b''8 r |
  }
  \new Staff = "left" {
    \clef bass
    s1
  }
>>

```



A glissando can connect notes in chords. If anything other than a direct one-to-one pairing of the notes in the two chords is required the connections between the notes are defined by setting `\glissandoMap`, where the notes of a chord are assumed to be numbered from zero in the order in which they appear in the input `.ly` file.

```
\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'> |
}
```



Different styles of glissandi can be created. For details, see Secció 5.4.8 [Line styles], pàgina 616.

Fragments de codi seleccionats

Contemporary glissando

A contemporary glissando without a final note can be typeset using a hidden note and cadenza timing.

```
\relative c'' {
  \time 3/4
  \override Glissando.style = #'zigzag
  c4 c
  \cadenzaOn
  c4\glissando
  \hideNotes
  c,,4
  \unHideNotes
  \cadenzaOff
}
```

```
\bar "|"
}
```



Adding timing marks to long glissandi

Skipped beats in very long glissandi are sometimes indicated by timing marks, often consisting of stems without noteheads. Such stems can also be used to carry intermediate expression markings.

If the stems do not align well with the glissando, they may need to be repositioned slightly.

```
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 |
}
```

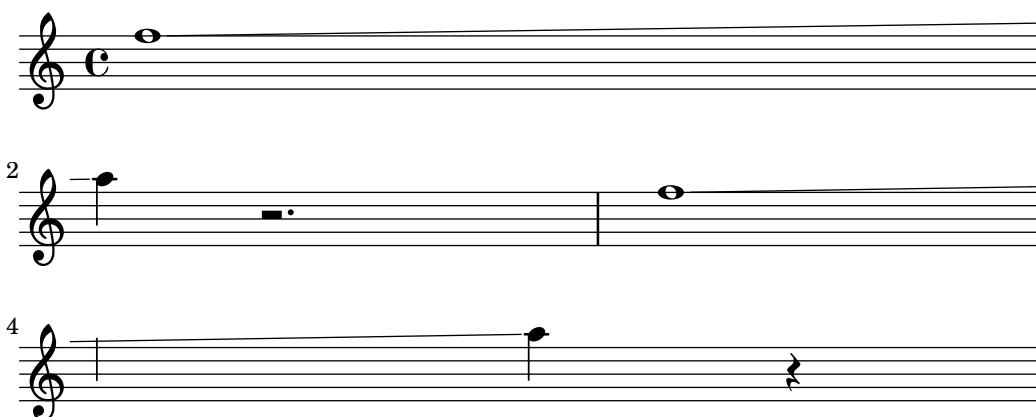


Making glissandi breakable

Setting the `breakable` property to `#t` in combination with `after-line-breaking` allows a glissando to break if it occurs at a line break:

```
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 |
}
```

*Extending glissandi across repeats*

A glissando which extends into several `\alternative` blocks can be simulated by adding a hidden grace note with a glissando at the start of each `\alternative` block. The grace note should be at the same pitch as the note which starts the initial glissando. This is implemented here with a music function which takes the pitch of the grace note as its argument.

Note that in polyphonic music the grace note must be matched with corresponding grace notes in all other voices.

```
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 }
    >>
  >>
}

```

Snippets: Secció “Expressive marks” in *Fragments de codi*.

Internals Reference: Secció “Glissando” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Printing text over the line (such as *gliss.*) is not supported.

Arpeggio

An *arpeggio* on a chord (also known as a broken chord) is denoted by appending `\arpeggio` to the chord construct:

```
\relative { <c' e g c>1\arpeggio }
```



Different types of arpeggios may be written. `\arpeggioNormal` reverts to a normal arpeggio:

```
\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
}
```



Special *bracketed* arpeggio symbols can be created:

```
\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
}
```




The dash properties of the parenthesis arpeggio are controlled with the 'dash-definition' property, which is described at [Slurs], pàgina 130.

Arpeggios can be explicitly written out with ties. For more information, see [Ties], pàgina 53.

Instruccions predefinides

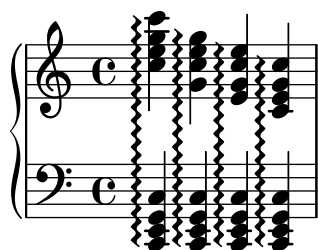
`\arpeggio`, `\arpeggioArrowUp`, `\arpeggioArrowDown`, `\arpeggioNormal`, `\arpeggioBracket`, `\arpeggioParenthesis`, `\arpeggioParenthesisDashed`.

Fragments de codi seleccionats

Creating cross-staff arpeggios in a piano staff

In a `PianoStaff`, it is possible to let an arpeggio cross between the staves by setting the property `PianoStaff.connectArpeggios`.

```
\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
    }
  }
>>
```



Creating cross-staff arpeggios in other contexts

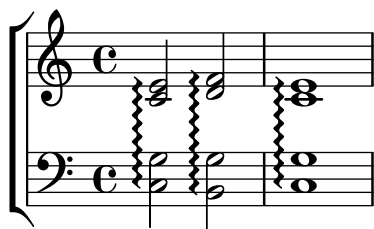
Cross-staff arpeggios can be created in contexts other than `GrandStaff`, `PianoStaff` and `StaffGroup` if the `Span_arpeggio_engraver` is included in the `Score` context.

```
\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"
  }
}
}

```



Creating arpeggios across notes in different voices

An arpeggio can be drawn across notes in different voices on the same staff if the `Span_arpeggio_engraver` is added to the `Staff` context:

```

\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 }
  >>
}

```



Vegeu també

Music Glossary: Secció “arpeggio” in *Glossari musical*.

Notation Reference: [Slurs], pàgina 130, [Ties], pàgina 53.

Snippets: Secció “Expressive marks” in *Fragments de codi*.

Internals Reference: Secció “Arpeggio” in *Referència de funcionament intern*, Secció “Slur” in *Referència de funcionament intern*, Secció “PianoStaff” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

It is not possible to mix connected arpeggios and unconnected arpeggios in one `PianoStaff` at the same point in time.

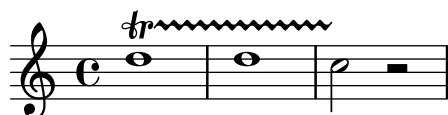
The simple way of setting parenthesis-style arpeggio brackets does not work for cross-staff arpeggios; see [Cross-staff stems], pàgina 323.

Trills

Short trills without an extender line are printed with `\trill`; see [Articulations and ornaments], pàgina 119.

Longer trills with an extender line are made with `\startTrillSpan` and `\stopTrillSpan`:

```
\relative {
  d''1\startTrillSpan
  d1
  c2\stopTrillSpan
  r2
}
```



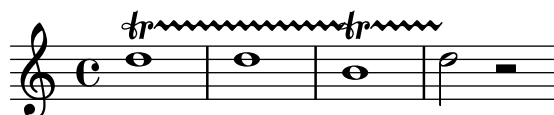
A trill spanner crossing a line break will restart exactly above the first note on the new line.

```
\relative {
  d''1\startTrillSpan
  \break
  d1
  c2\stopTrillSpan
  r2
}
```



Consecutive trill spans will work without explicit `\stopTrillSpan` commands, since successive trill spanners will automatically become the right bound of the previous trill.

```
\relative {
  d''1\startTrillSpan
  d1
  b1\startTrillSpan
  d2\stopTrillSpan
  r2
}
```



Trills can also be combined with grace notes. The syntax of this construct and the method to precisely position the grace notes are described in [Grace notes], pàgina 112.

```
\relative {
  d''1~\afterGrace
  d1\startTrillSpan { c32[ d]\stopTrillSpan }
  c2 r2
}
```



Trills that require an auxiliary note with an explicit pitch can be typeset with the `\pitchedTrill` command. The first argument is the main note, and the second is the *trilled* note, printed as a stemless note head in parentheses.

```
\relative {
  \pitchedTrill
  d''2\startTrillSpan fis
  d2
  c2\stopTrillSpan
  r2
}
```



Subsequent accidentals of the same note in the same measure will need to be added manually. Only the accidental of the first pitched trill in a measure is printed.

```
\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
}
```



Instruccions predefinides

`\startTrillSpan`, `\stopTrillSpan`.

Vegeu també

Music Glossary: Secció “trill” in *Glossari musical*.

Notation Reference: [Articulations and ornamentations], pàgina 119, [Grace notes], pàgina 112.

Snippets: Secció “Expressive marks” in *Fragments de codi*.

Internals Reference: Secció “TrillSpanner” in *Referència de funcionament intern*.

1.4 Repeats



Repetition is a central concept in music, and multiple notations exist for repetitions. LilyPond supports the following kinds of repeats:

- volta** The repeated music is not written out but enclosed between repeat bar lines. If the repeat is at the beginning of a piece, a repeat bar line is only printed at the end of the repeat. Alternative endings (volte) are printed left to right with brackets. This is the standard notation for repeats with alternatives.
- unfold** The repeated music is fully written out, as many times as specified by *repeatcount*. This is useful when entering repetitious music.
- percent** These are beat or measure repeats. They look like single slashes or percent signs.
- tremolo** This is used to write tremolo beams.

1.4.1 Long repeats

This section discusses how to input long (usually multi-measure) repeats. The repeats can take two forms: repeats enclosed between repeat signs; or written-out repeats, used to input repetitious music. Repeat signs can also be controlled manually.

Normal repeats

The syntax for a normal repeat is

```
\repeat volta repeatcount musicexpr
```

where *musicexpr* is a music expression.

A single repeat without an alternate ending:

```
\relative {
  \repeat volta 2 { c''4 d e f }
  c2 d
  \repeat volta 2 { d4 e f g }
}
```



An ‘opening’ repeat mark is not, by default, printed in the first full measure. However it is possible to add one by using `\bar ".|:"` before the first note.

```
\relative {
  \repeat volta 2 { \bar ".|:" c''4 d e f }
  c2 d
  \repeat volta 2 { d4 e f g }
}
```



Alternative endings can be produced using `\alternative`. Each group of alternatives must be themselves, enclosed in a set of braces.

```
\repeat volta repeatcount musicexpr
\alternative {
  { musicexpr }
}
```

where *musicexpr* is a music expression.

If there are more repeats than there are alternate endings, the earliest repeats are given the first alternative.

A single repeat with one alternate ending:

```
\relative {
  \repeat volta 2 { c''4 d e f | }
  \alternative {
    { c2 e | }
    { f2 g | }
  }
  c1
}
```



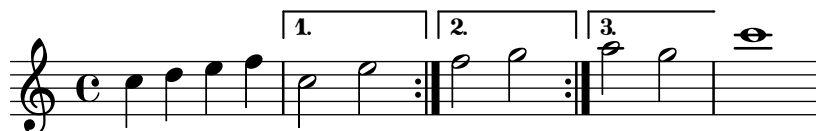
Multiple repeats with one alternate ending:

```
\relative {
  \repeat volta 4 { c''4 d e f | }
  \alternative {
    { c2 e | }
    { f2 g | }
  }
  c1
}
```



Multiple repeats with more than one alternate ending:

```
\relative {
  \repeat volta 3 { c''4 d e f | }
  \alternative {
    { c2 e | }
    { f2 g | }
    { a2 g | }
  }
  c1
}
```



Nota: If there are two or more alternatives, nothing should appear between the closing brace of one and the opening brace of the next in an `\alternative` block, otherwise you will not get the expected number of endings.

Nota: If you include `\relative` inside a `\repeat` without explicitly instantiating the `Voice` context, extra (unwanted) staves will appear. See Secció “An extra staff appears” in *Utilització del programa*.

If a repeat that has no alternate endings starts in the middle of a measure, it will usually end at a corresponding place in the middle of a later measure (so that the two ends add up to one complete measure). In this case the repeat signs are not ‘true’ bar lines so neither bar checks nor `\partial` commands should be placed there:

```
c'4 e g
\repeat volta 4 {
  e4 |
  c2 e |
  g4 g g
}
g4 |
```

```
a2 a |
g1 |
```



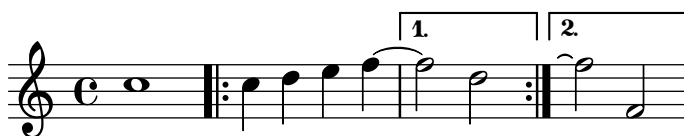
If a repeat that has no alternate endings starts with a partial measure, then the same principles apply, except that a `\partial` command is required at the start of the measure:

```
\partial 4
\repeat volta 4 {
  e'4 |
  c2 e |
  g4 g g
}
g4 |
a2 a |
g1 |
```



Ties may be added to a second ending:

```
\relative {
  c''1
  \repeat volta 2 { c4 d e f~ }
  \alternative {
    { f2 d }
    { f2\repeatTie f, }
  }
}
```



The `\inStaffSegno` command can be used to generate a composite bar line incorporating the segno symbol with the appropriate repeat bar line when used with the `\repeat volta` command. The correct type of repeat bar line, viz. start repeat, end repeat or double repeat, is selected automatically. Note that the corresponding “D.S.” mark must be added manually.

Away from a repeat:

```
\relative {
  e'1
  \inStaffSegno
  f2 g a b
  c1_"D.S." \bar " | ."
}
```



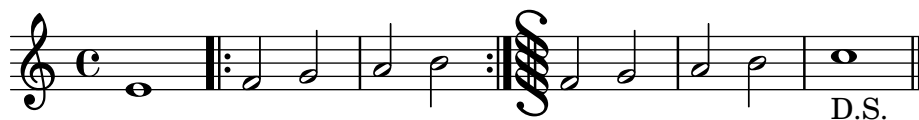
At the start of a repeat:

```
\relative {
  e'1
  \repeat volta 2 {
    \inStaffSegno % start repeat
    f2 g a b
  }
  c1_"D.S." \bar "|."
}
```



At the end of a repeat:

```
\relative {
  e'1
  \repeat volta 2 {
    f2 g a b
    \inStaffSegno % end repeat
  }
  f2 g a b
  c1_"D.S." \bar "|."
}
```



Between two repeats:

```
\relative {
  e'1
  \repeat volta 2 {
    f2 g a b
  }
  \inStaffSegno % double repeat
  \repeat volta 2 {
    f2 g a b
  }
  c1_"D.S." \bar "|."
}
```



Alternative bar line symbols can be obtained by setting (in the Score context) the properties `segnoType`, `startRepeatSegnoType`, `endRepeatSegnoType` or `doubleRepeatSegnoType` to the required bar line type. The alternative bar line types must be selected from the pre-defined types or types previously defined with the `\defineBarLine` command (see [Bar lines], pàgina 97).

```
\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 "|."
}

```



Fragments de codi seleccionats

Shortening volta brackets

By default, the volta brackets will be drawn over all of the alternative music, but it is possible to shorten them by setting `voltaSpannerDuration`. In the next example, the bracket only lasts one measure, which is a duration of $3/4$.

```

\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 }
  }
}

```



Adding volta brackets to additional staves

The `Volta_engraver` by default resides in the `Score` context, and brackets for the repeat are thus normally only printed over the topmost staff. This can be adjusted by adding the `Volta_engraver` to the `Staff` context where the brackets should appear; see also the “Volta multi staff” snippet.

```

<<
  \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' } }
>>

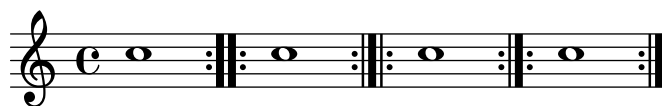
```



Setting the double repeat default for volte

There are three different styles of double repeats for volte, that can be set using `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 }
}
```



Alternative bar numbering

Two alternative methods for bar numbering can be set, especially for when using repeated music.

```
\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 six musical staves, each representing a different type of musical ending. The first staff shows a first ending (marked '1.') that repeats a sequence of notes. The second staff shows a second ending (marked '2.') that also repeats a sequence of notes. The third staff shows a third ending (marked '3.') that repeats a sequence of notes. The fourth staff shows a first ending (marked '1.') that repeats a sequence of notes. The fifth staff shows a second ending (marked '2.') that repeats a sequence of notes. The sixth staff shows a third ending (marked '3.') that repeats a sequence of notes. Each staff is written on a five-line staff with a treble clef and a common time signature (C).

Vegeu també

Music Glossary: Secció “repeat” in *Glossari musical*, Secció “volta” in *Glossari musical*.

Notation Reference: [Bar lines], pàgina 97, Secció 5.1.4 [Modifying context plug-ins], pàgina 579, [Modifying ties and slurs], pàgina 624, [Time administration], pàgina 118.

Installed Files: `ly/engraver-init.ly`.

Snippets: Secció “Repeats” in *Fragments de codi*.

Internals Reference: Secció “VoltaBracket” in *Referència de funcionament intern*, Secció “RepeatedMusic” in *Referència de funcionament intern*, Secció “VoltaRepeatedMusic” in *Referència de funcionament intern*, Secció “UnfoldedRepeatedMusic” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Slurs that span from a `\repeat` block into an `\alternative` block will only work for the first alternative ending. The visual appearance of a continuing slur in other alternative blocks may be simulated with `\repeatTie` if the slur extends into only one note in the alternative block, although this method does not work in `TabStaff`. Other methods which may be tailored to indicate continuing slurs over several notes in alternative blocks, and which also work in `TabStaff` contexts, are shown in [Modifying ties and slurs], pàgina 624.

Also, slurs cannot wrap around from the end of one alternative back to the beginning of the repeat.

Glissandi that span from a `\repeat` block into an `\alternative` block will only work for the first alternative ending. The visual appearance of a continuing glissando in other alternative blocks may be indicated by coding a glissando starting on a hidden grace note. For an example, see “Extending glissandi across repeats” under Selected Snippets in [Glissando], pàgina 137.

If a repeat that begins with an incomplete measure has an `\alternative` block that contains modifications to the `measureLength` property, using `\unfoldRepeats` will result in wrongly-placed bar lines and bar check warnings.

A nested repeat like

```
\repeat ...
\repeat ...
\alternative
```

is ambiguous, since it is not clear to which `\repeat` the `\alternative` belongs. This ambiguity is resolved by always having the `\alternative` belong to the inner `\repeat`. For clarity, it is advisable to use braces in such situations.

Manual repeat marks

Nota: These methods are only used for displaying unusual repeat constructs, and may produce unexpected behavior. In most cases, repeats should be created using the standard `\repeat` command or by printing the relevant bar lines. For more information, see [Bar lines], pàgina 97.

The property `repeatCommands` can be used to control the layout of repeats. Its value is a Scheme list of repeat commands.

start-repeat

Print a `.|:` bar line.

```
\relative {
  c''1
  \set Score.repeatCommands = #'(start-repeat)
  d4 e f g
  c1
}
```



As per standard engraving practice, repeat signs are not printed at the beginning of a piece.

end-repeat

Print a `:|.` bar line:

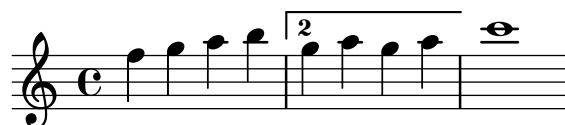
```
\relative {
  c''1
  d4 e f g
  \set Score.repeatCommands = #'(end-repeat)
  c1
}
```



(volta *number*) ... (volta *#f*)

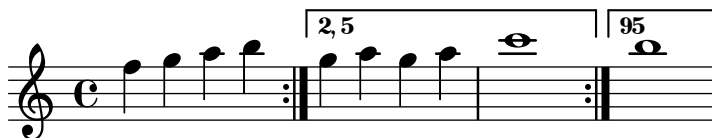
Create a new volta with the specified number. The volta bracket must be explicitly terminated, or it will not be printed.

```
\relative {
  f''4 g a b
  \set Score.repeatCommands = #'((volta "2"))
  g4 a g a
  \set Score.repeatCommands = #'((volta #f))
  c1
}
```



Multiple repeat commands may occur at the same point:

```
\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))
}
```



Text can be included with the volta bracket. The text can be a number or numbers or markup text, see Secció 1.8.2 [Formatting text], pàgina 237. The simplest way to use markup text is to define the markup first, then include the markup in a Scheme list.

```
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))
}
```



Vegeu també

Notation Reference: [Bar lines], pàgina 97, Secció 1.8.2 [Formatting text], pàgina 237.

Snippets: Secció “Repeats” in *Fragments de codi*.

Internals Reference: Secció “VoltaBracket” in *Referència de funcionament intern*, Secció “RepeatedMusic” in *Referència de funcionament intern*, Secció “VoltaRepeatedMusic” in *Referència de funcionament intern*.

Written-out repeats

By using the `unfold` command, repeats can be used to simplify the writing out of repetitious music. The syntax is

```
\repeat unfold repeatcount musicexpr
```

where *musicexpr* is a music expression and *repeatcount* is the number of times *musicexpr* is repeated.

```
\relative {
  \repeat unfold 2 { c''4 d e f }
  c1
}
```



In some cases, especially in a `\relative` context, the `\repeat unfold` function is not the same as writing out the music expression multiple times. E.g,

```
\repeat unfold 2 { a'4 b c }
```

is not equivalent to

```
a'4 b c | a'4 b c
```

Unfold repeats can be made with alternate endings.

```
\relative {
  \repeat unfold 2 { c''4 d e f }
  \alternative {
    { c2 g' }
    { c,2 b }
  }
  c1
}
```



If there are more repeats than there are alternate endings, the first alternative is applied multiple times until the remaining alternatives make up the total number of repeats.

```
\relative {
  \repeat unfold 4 { c''4 d e f }
  \alternative {
    { c2 g' }
    { c,2 b }
    { e2 d }
  }
```

```

    }
  c1
}

```



If there are more alternate endings than repeats then only the first alternatives are applied. The remaining alternatives will be ignored and not printed.

```

\relative {
  \repeat unfold 2 { c''4 d e f }
  \alternative {
    { c2 g' }
    { c,2 b }
    { e2 d }
  }
  c1
}

```



It is also possible to nest multiple `unfold` functions (with or without alternate endings).

```

\relative {
  \repeat unfold 2 {
    \repeat unfold 2 { c''4 d e f }
    \alternative {
      { c2 g' }
      { c,2 b }
    }
  }
  c1
}

```



Chord constructs can be repeated by the chord repetition symbol `q`. See [Chord repetition], pàgina 166.

Nota: If you include `\relative` inside a `\repeat` without explicitly instantiating the `Voice` context, extra (unwanted) staves will appear. See Secció “An extra staff appears” in *Utilització del programa*.

Vegeu també

Notation Reference: [Chord repetition], pàgina 166.

Snippets: Secció “Repeats” in *Fragments de codi*.

Internals Reference: Secció “RepeatedMusic” in *Referència de funcionament intern*, Secció “UnfoldedRepeatedMusic” in *Referència de funcionament intern*.

1.4.2 Short repeats

This section discusses how to input short repeats. Short repeats can take two forms: slashes or percent signs to represent repeats of a single note, a single measure or two measures, and tremolos otherwise.

Percent repeats

Repeated short patterns are printed once, and the repeated pattern is replaced with a special sign.

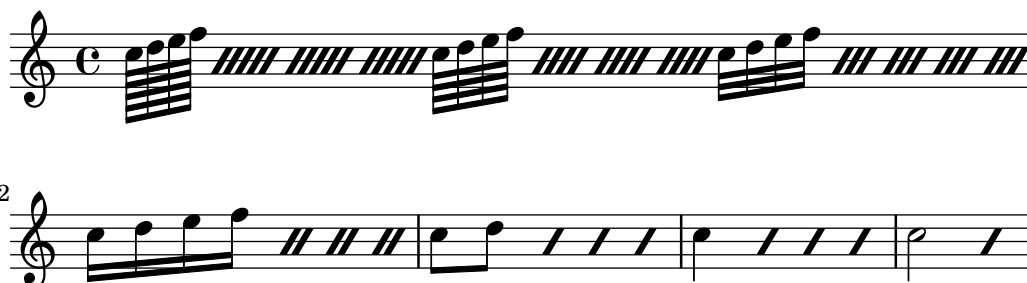
The syntax is

```
\repeat percent number musicexpr
```

where *musicexpr* is a music expression.

Patterns that are shorter than one measure are replaced by slashes.

```
\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 }
}
```



Patterns of one or two measures are replaced by percent-like symbols.

```
\relative c'' {
  \repeat percent 2 { c4 d e f }
  \repeat percent 2 { c2 d }
  \repeat percent 2 { c1 }
}
```

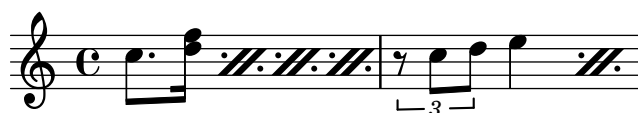


```
\relative {
  \repeat percent 3 { c''4 d e f | c2 g' }
}
```



Patterns that are shorter than one measure but contain mixed durations use a double-percent symbol.

```
\relative {
  \repeat percent 4 { c''8. <d f>16 }
  \repeat percent 2 { \tuplet 3/2 { r8 c d } e4 }
}
```



Fragments de codi seleccionats

Percent repeat counter

Measure repeats of more than two repeats can get a counter when the convenient property is switched, as shown in this example:

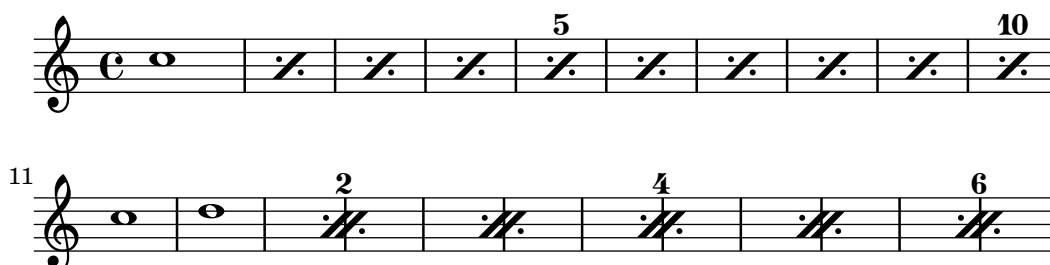
```
\relative c'' {
  \set countPercentRepeats = ##t
  \repeat percent 4 { c1 }
}
```



Percent repeat count visibility

Percent repeat counters can be shown at regular intervals by setting the context property `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 }
}
```



Isolated percent repeats

Isolated percents can also be printed.

```
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
}
```



Vegeu també

Music Glossary: Secció “percent repeat” in *Glossari musical*, Secció “simile” in *Glossari musical*.

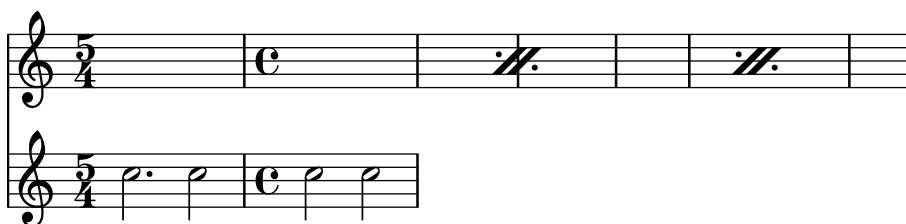
Snippets: Secció “Repeats” in *Fragments de codi*.

Internals Reference: Secció “RepeatSlash” in *Referència de funcionament intern*, Secció “RepeatSlashEvent” in *Referència de funcionament intern*, Secció “DoubleRepeatSlash” in *Referència de funcionament intern*, Secció “PercentRepeat” in *Referència de funcionament intern*, Secció “PercentRepeatCounter” in *Referència de funcionament intern*, Secció “PercentRepeatedMusic” in *Referència de funcionament intern*, Secció “Percent_repeat_engraver” in *Referència de funcionament intern*, Secció “DoublePercentEvent” in *Referència de funcionament intern*, Secció “DoublePercentRepeat” in *Referència de funcionament intern*, Secció “DoublePercentRepeatCounter” in *Referència de funcionament intern*, Secció “Double_percent_repeat_engraver” in *Referència de funcionament intern*, Secció “Slash_repeat_engraver” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

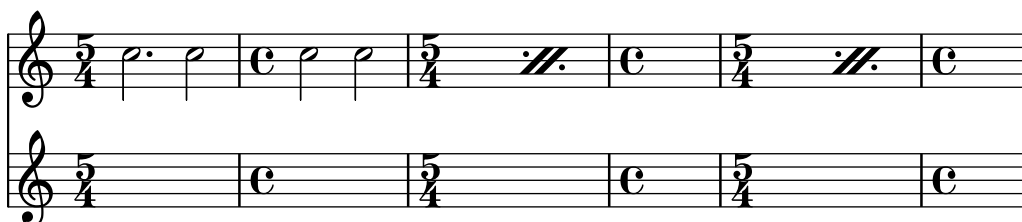
Percent repeats will not contain anything else apart from the percent sign itself; in particular, timing changes will not be repeated.

```
\repeat percent 3 { \time 5/4 c2. 2 \time 4/4 2 2 }
```



Any meter changes or `\partial` commands need to occur in parallel passages *outside* of any percent repeat, e.g in a separate timing track.

```
<<
  \repeat percent 3 { c2. 2 2 2 }
  \repeat unfold 3 { \time 5/4 s4*5 \time 4/4 s1 }
>>
```

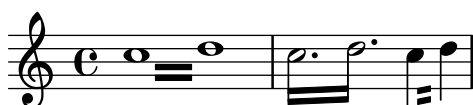


Tremolo repeats

Tremolos can take two forms: alternation between two chords or two notes, and rapid repetition of a single note or chord. Tremolos consisting of an alternation are indicated by adding beams between the notes or chords being alternated, while tremolos consisting of the rapid repetition of a single note are indicated by adding beams or slashes to a single note.

To place tremolo marks between notes, use `\repeat` with tremolo style:

```
\relative c'' {
  \repeat tremolo 8 { c16 d }
  \repeat tremolo 6 { c16 d }
  \repeat tremolo 2 { c16 d }
}
```



The `\repeat tremolo` syntax expects exactly two notes within the braces, and the number of repetitions must correspond to a note value that can be expressed with plain or dotted notes. Thus, `\repeat tremolo 7` is valid and produces a double dotted note, but `\repeat tremolo 9` is not.

The duration of the tremolo equals the duration of the braced expression multiplied by the number of repeats: `\repeat tremolo 8 { c16 d16 }` gives a whole note tremolo, notated as two whole notes joined by tremolo beams.

There are two ways to put tremolo marks on a single note. The `\repeat tremolo` syntax is also used here, in which case the note should not be surrounded by braces:

```
\repeat tremolo 4 c'16
```



The same output can be obtained by adding `:N` after the note, where N indicates the duration of the subdivision (it must be at least 8). If N is 8, one beam is added to the note's stem. If N is omitted, the last value is used:

```
\relative {
  c'12:8 c:32
  c: c:
}
```



Fragments de codi seleccionats

Cross-staff tremolos

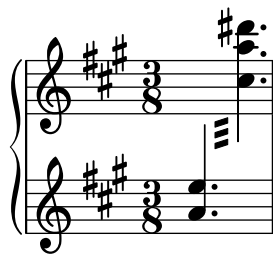
Since `\repeat tremolo` expects exactly two musical arguments for chord tremolos, the note or chord which changes staff within a cross-staff tremolo should be placed inside curly braces together with its `\change Staff` command.

```
\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
    }
  }
}
}
>>

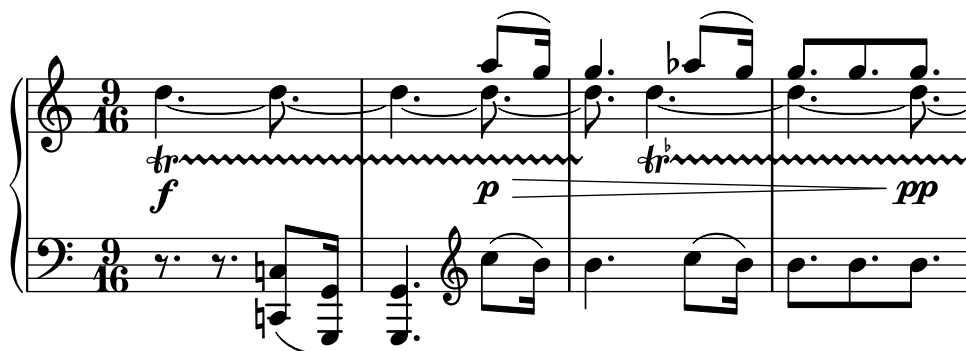
```



Vegeu també

Snippets: Secció “Repeats” in *Fragments de codi*.

1.5 Simultaneous notes



Polyphony in music refers to having more than one voice occurring in a piece of music. Polyphony in LilyPond refers to having more than one voice on the same staff.

1.5.1 Single voice

This section discusses simultaneous notes inside the same voice.

Chorded notes

A chord is formed by enclosing a set of pitches between `<` and `>`. A chord may be followed by a duration just like simple notes.

```
\relative {
  <a' c e>1 <a c e>2 <f a c e>4 <a c>8. <g c e>16
}
```

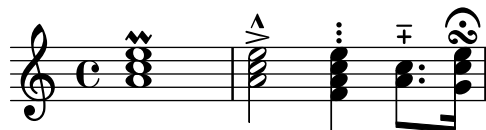
Chords may also be followed by articulations, again just like simple notes.

```
\relative {
  <a' c e>1\fermata <a c e>2-> <f a c e>4\prall <a c>8.^! <g c e>16-.
}
```

The notes within the chord themselves can also be followed by articulation and ornamentation.

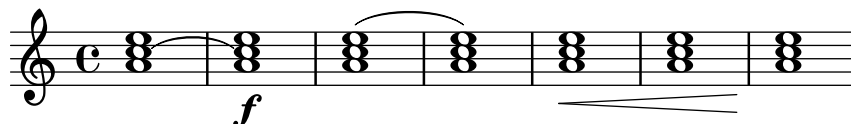
```
\relative {
  <a' c>\prall e>1 <a-> c-^ e>2 <f-. a c-. e-.>4
}
```

```
<a+ c-->8. <g\fermata c e\turn>16
}
```



However some notation, such as dynamics and hairpins must be attached to the chord rather than to notes within the chord, otherwise they will not print. Other notation like fingerings and slurs will get placed markedly different when attached to notes within a chord rather than to whole chords or single notes.

```
\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>\!
}
```



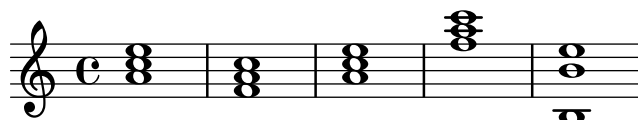
A chord acts merely as a container for its notes, its articulations and other attached elements. Consequently, a chord without notes inside does not actually have a duration. Any attached articulations will happen at the same musical time as the next following note or chord and be combined with them (for more complex possibilities of combining such elements, see [Simultaneous expressions], pàgina 168):

```
\relative {
  \grace { g'8( a b }
  <> ) \p \< -. -\markup \italic "sempre staccato"
  \repeat unfold 4 { c4 e } c1\f
}
```



Relative mode can be used for pitches in chords. The first note of each chord is always relative to the first note of the chord that came before it, or in the case where no preceding chord exists, the pitch of the last note that came before the chord. All remaining notes in the chord are relative to the note that came before it *within the same chord*.

```
\relative {
  <a' c e>1 <f a c> <a c e> <f' a c> <b, e b,>
}
```



For more information about chords, see Secció 2.7 [Chord notation], pàgina 404.

Vegeu també

Music Glossary: Secció “chord” in *Glossari musical*.

Learning Manual: Secció “Combining notes into chords” in *Manual d’aprenentatge*.

Notation Reference: Secció 2.7 [Chord notation], pàgina 404, [Articulations and ornamentations], pàgina 119, (undefined) [Relative octave entry], pàgina (undefined), Secció 1.5.2 [Multiple voices], pàgina 169.

Snippets: Secció “Simultaneous notes” in *Fragments de codi*.

Advertiments i problemes coneguts

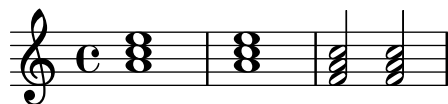
Chords containing more than two pitches within a staff space, such as ‘<e f! fis!>’, create overlapping noteheads. Depending on the situation, better representations might involve

- temporary use of Secció 1.5.2 [Multiple voices], pàgina 169, ‘<< f! \\ <e fis!> >>’,
- enharmonic transcription of one or more pitches, ‘<e f ges>’, or
- [Clusters], pàgina 169.

Chord repetition

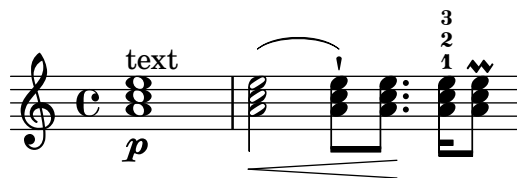
In order to save typing, a shortcut can be used to repeat the preceding chord. The chord repetition symbol is q:

```
\relative {
  <a' c e>1 q <f a c>2 q
}
```



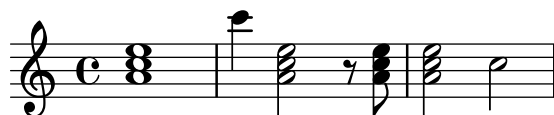
As with regular chords, the chord repetition symbol can be used with durations, articulations, markups, slurs, beams, etc., as only the pitches of the previous chord are duplicated.

```
\relative {
  <a' c e>1\p~"text" q2\<( q8)[-! q8.]!\! q16-1-2-3 q8\prall
}
```



The chord repetition symbol always remembers the last instance of a chord so it is possible to repeat the most recent chord even if other non-chorded notes or rests have been added since.

```
\relative {
  <a' c e>1 c'4 q2 r8 q8 |
  q2 c, |
}
```



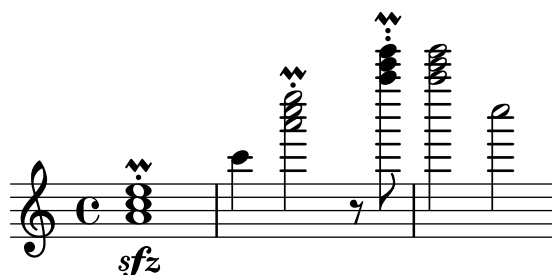
However, the chord repetition symbol does not retain any dynamics, articulation or ornamentation within, or attached to, the previous chord.

```
\relative {
  <a'-. c\prall e>1\s fz c'4 q2 r8 q8 |
  q2 c, |
}
```



To have some of them retained, the `\chordRepeats` function can be called explicitly with an extra argument specifying a list of *event types* to keep unless events of that type are already present on the `q` chord itself.

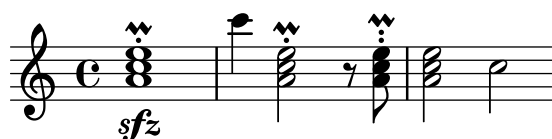
```
\relative {
  \chordRepeats #'(articulation-event)
  { <a'-. c\prall e>1\s fz c'4 q2 r8 q8-. } |
  q2 c, |
}
```



Here using `\chordRepeats` inside of a `\relative` construction produces unexpected results: once chord events have been expanded, they are indistinguishable from having been entered as regular chords, making `\relative` assign an octave based on their current context.

Since nested instances of `\relative` don't affect one another, another `\relative` inside of `\chordRepeats` can be used for establishing the octave relations before expanding the repeat chords. In that case, the whole content of the inner `\relative` does not affect the outer one; hence the different octave entry of the final note in this example.

```
\relative {
  \chordRepeats #'(articulation-event)
  \relative
  { <a'-. c\prall e>1\s fz c'4 q2 r8 q8-. } |
  q2 c' ' |
}
```



Interactions with `\relative` occur only with explicit calls of `\chordRepeats`: the implicit expansion at the start of typesetting is done at a time where all instances of `\relative` have already been processed.

Vegeu també

Notation Reference: Secció 2.7 [Chord notation], pàgina 404, [Articulations and ornamentations], pàgina 119.

Installed Files: `ly/chord-repetition-init.ly`.

Simultaneous expressions

One or more music expressions enclosed in double angle brackets are taken to be simultaneous. If the first expression begins with a single note or if the whole simultaneous expression appears explicitly within a single voice, the whole expression is placed on a single staff; otherwise the elements of the simultaneous expression are placed on separate staves.

The following examples show simultaneous expressions on one staff:

```
\new Voice { % explicit single voice
  << \relative { a'4 b g2 }
    \relative { d'4 g c,2 } >>
}
```



```
\relative {
  % single first note
  a' << \relative { a'4 b g }
    \relative { d'4 g c, } >>
}
```



This can be useful if the simultaneous sections have identical rhythms, but attempts to attach notes with different durations to the same stem will cause errors. Notes, articulations, and property changes in a *single* 'Voice' are collected and engraved in musical order:

```
\relative {
  <a' c>4-. <>-. << c a >> << { c-. <c a> } { a s-. } >>
}
```



Multiple stems or beams or different note durations or properties at the same musical time require the use of multiple voices.

The following example shows how simultaneous expressions can generate multiple staves implicitly:

```
% no single first note
<< \relative { a'4 b g2 }
  \relative { d'4 g2 c,4 } >>
```



Here different rhythms cause no problems because they are interpreted in different voices.

Advertiments i problemes coneguts

If notes from two or more voices, with no shifts specified, have stems in the same direction, the message

`warning: This voice needs a \voiceXx or \shiftXx setting`

will appear during compilation. This message can be suppressed by:

`\override NoteColumn.ignore-collision = ##t`

However, this not only suppresses the warning but will prevent any collision resolution whatsoever and may have other unintended effects (also see *Known Issues* in [Collision resolution], pàgina 173).

Clusters

A cluster indicates a continuous range of pitches to be played. They can be denoted as the envelope of a set of notes. They are entered by applying the function `\makeClusters` to a sequence of chords, e.g.,

`\relative \makeClusters { <g' b>2 <c g'> }`



Ordinary notes and clusters can be put together in the same staff, even simultaneously. In such a case no attempt is made to automatically avoid collisions between ordinary notes and clusters.

Vegeu també

Music Glossary: Secció “cluster” in *Glossari musical*.

Snippets: Secció “Simultaneous notes” in *Fragments de codi*.

Internals Reference: Secció “ClusterSpanner” in *Referència de funcionament intern*, Secció “ClusterSpannerBeacon” in *Referència de funcionament intern*, Secció “Cluster_spanner_engraver” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Clusters look good only if they span at least two chords; otherwise they appear too narrow.

Clusters do not have a stem and cannot indicate durations by themselves, but the length of the printed cluster is determined by the durations of the defining chords. Separate clusters need a separating rest between them.

Clusters do not produce MIDI output.

1.5.2 Multiple voices

This section discusses simultaneous notes in multiple voices or multiple staves.

Single-staff polyphony

Explicitly instantiating voices

The basic structure needed to achieve multiple independent voices in a single staff is illustrated in the following example:

```
\new Staff <<
  \new Voice = "first"
    \relative { \voiceOne r8 r16 g'' e8. f16 g8[ c,] f e16 d }
  \new Voice= "second"
    \relative { \voiceTwo d''16 c d8~ 16 b c8~ 16 b c8~ 16 b8. }
>>
```



Here, voices are instantiated explicitly and are given names. The `\voiceOne ... \voiceFour` commands set up the voices so that first and third voices get stems up, second and fourth voices get stems down, third and fourth voice note heads are horizontally shifted, and rests in the respective voices are automatically moved to avoid collisions. The `\oneVoice` command returns all the voice settings to the neutral default directions.

Temporary polyphonic passages

A temporary polyphonic passage can be created with the following construct:

```
<< { \voiceOne ... }
  \new Voice { \voiceTwo ... }
>> \oneVoice
```

Here, the first expression within a temporary polyphonic passage is placed into the `Voice` context which was in use immediately before the polyphonic passage, and that same `Voice` context continues after the temporary section. Other expressions within the angle brackets are assigned to distinct temporary voices. This allows lyrics to be assigned to one continuing voice before, during and after a polyphonic section:

```
\relative <<
  \new Voice = "melody" {
    a'4
    <<
      {
        \voiceOne
        g f
      }
      \new Voice {
        \voiceTwo
        d2
      }
    >>
    \oneVoice
    e4
  }
  \new Lyrics \lyricsto "melody" {
    This is my song.
  }
>>
```



Here, the `\voiceOne` and `\voiceTwo` commands are required to define the settings of each voice.

The double backslash construct

The `<< {...} \ \ {...} >>` construct, where the two (or more) expressions are separated by double backslashes, behaves differently to the similar construct without the double backslashes: *all* the expressions within this construct are assigned to new **Voice** contexts. These new **Voice** contexts are created implicitly and are given the fixed names "1", "2", etc.

The first example could be typeset as follows:

```
<<
  \relative { r8 r16 g'' e8. f16 g8[ c,] f e16 d }
  \ \
  \relative { d''16 c d8~ 16 b c8~ 16 b c8~ 16 b8. }
>>
```



This syntax can be used where it does not matter that temporary voices are created and then discarded. These implicitly created voices are given the settings equivalent to the effect of the `\voiceOne ... \voiceFour` commands, in the order in which they appear in the code.

In the following example, the intermediate voice has stems up, therefore we enter it in the third place, so it becomes voice three, which has the stems up as desired. Spacer rests are used to avoid printing doubled rests.

```
<<
  \relative { r8 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 all but the simplest works it is advisable to create explicit **Voice** contexts as explained in Secció “Contexts and engravers” in *Manual d’aprenentatge* and Secció “Explicitly instantiating voices” in *Manual d’aprenentatge*.

Voice order

When entering multiple voices in the input file, use the following order:

```
Voice 1: highest
Voice 2: lowest
Voice 3: second highest
```

Voice 4: second lowest
 Voice 5: third highest
 Voice 6: third lowest
 etc.

Though this may seem counterintuitive, it simplifies the automatic layout process. Note that the odd-numbered voices are given upstems, and the even-numbered voices are given downstems:

```
\new Staff <<
  \time 2/4
  { f''2 } % 1: highest
  \\
  { c'2 } % 2: lowest
  \\
  { d''2 } % 3: second-highest
  \\
  { e'2 } % 4: second-lowest
  \\
  { b'2 } % 5: third-highest
  \\
  { g'2 } % 6: third-lowest
>>
```



Nota: Lyrics and spanners (such as slurs, ties, hairpins, etc.) cannot be created ‘across’ voices.

Identical rhythms

In the special case that we want to typeset parallel pieces of music that have the same rhythm, we can combine them into a single **Voice** context, thus forming chords. To achieve this, enclose them in a simple simultaneous music construct within an explicit voice:

```
\new Voice <<
  \relative { e''4 f8 d e16 f g8 d4 }
  \relative { c''4 d8 b c16 d e8 b4 }
>>
```



This method leads to strange beamings and warnings if the pieces of music do not have the same rhythm.

Instruccions predefinides

\voiceOne, \voiceTwo, \voiceThree, \voiceFour, \oneVoice.

Vegeu també

Learning Manual: Secció “Voices contain music” in *Manual d’aprenentatge*, Secció “Explicitly instantiating voices” in *Manual d’aprenentatge*.

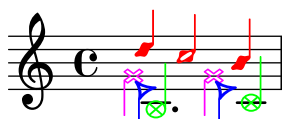
Notation Reference: [Percussion staves], pàgina 383, [Invisible rests], pàgina 59, [Stems], pàgina 224.

Snippets: Secció “Simultaneous notes” in *Fragments de codi*.

Voice styles

Voices may be given distinct colors and shapes, allowing them to be easily identified:

```
<<
  \relative { \voiceOneStyle d''4 c2 b4 }
  \\
  \relative { \voiceTwoStyle e'2 e }
  \\
  \relative { \voiceThreeStyle b2. c4 }
  \\
  \relative { \voiceFourStyle g'2 g }
>>
```



The `\voiceNeutralStyle` command is used to revert to the standard presentation.

Instruccions predefinides

`\voiceOneStyle`, `\voiceTwoStyle`, `\voiceThreeStyle`, `\voiceFourStyle`,
`\voiceNeutralStyle`.

Vegeu també

Learning Manual: Secció “I’m hearing Voices” in *Manual d’aprenentatge*, Secció “Other sources of information” in *Manual d’aprenentatge*.

Snippets: Secció “Simultaneous notes” in *Fragments de codi*.

Collision resolution

The note heads of notes in different voices with the same pitch, same note head and opposite stem direction are automatically merged, but notes with different note heads or the same stem direction are not. Rests opposite a stem in a different voice are shifted vertically. The following example shows three different circumstances, on beats 1 and 3 in bar 1 and beat 1 in bar 2, where the automatic merging fails.

```
<<
  \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
}
>>

```



Notes with different note heads may be merged as shown below. In this example the note heads on beat 1 of bar 1 are now merged:

```

<<
\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
}
>>

```



Quarter and half notes are not merged in this way, since it would be difficult to tell them apart.

Note heads with different dots as shown in beat 3 of bar 1 may be also be merged:

```

<<
\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
  }
>>

```



The half note and eighth note at the start of the second measure are incorrectly merged because the automatic merge cannot successfully complete the merge when three or more notes line up in the same note column, and in this case the merged note head is incorrect. To allow the merge to select the correct note head a `\shift` must be applied to the note that should not be merged. Here, `\shiftOn` is applied to move the top *g* out of the column, and `\mergeDifferentlyHeadedOn` then works properly.

```

<<
  \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
  }
>>

```



The `\shiftOn` command allows (but does not force) the notes in a voice to be shifted. When `\shiftOn` is applied to a voice, a note or chord in that voice is shifted only if its stem would otherwise collide with a stem from another voice, and only if the colliding stems point in the same direction. The `\shiftOff` command prevents this type of shifting from occurring.

By default, the outer voices (normally voices one and two) have `\shiftOff` specified, while the inner voices (three and above) have `\shiftOn` specified. When a shift is applied, voices with upstems (odd-numbered voices) are shifted to the right, and voices with downstems (even-numbered voices) are shifted to the left.

Here is an example to help you visualize how an abbreviated polyphonic expression would be expanded internally.

Nota: Note that with three or more voices, the vertical order of voices in your input file should not be the same as the vertical order of voices on the staff!

```
\new Staff \relative {
  %% abbreviated entry
  <<
    { f'2 } % 1: highest
    \\\
    { g,2 } % 2: lowest
    \\\
    { d'2 } % 3: upper middle
    \\\
    { b2 } % 4: lower middle
  >>
  %% internal expansion of the above
  <<
    \new Voice = "1" { \voiceOne \shiftOff f'2 }
    \new Voice = "2" { \voiceTwo \shiftOff g,2 }
    \new Voice = "3" { \voiceThree \shiftOn d'2 } % shifts right
    \new Voice = "4" { \voiceFour \shiftOn b2 } % shifts left
  >>
}
```



Two additional commands, `\shiftOnn` and `\shiftOnnn` provide further shift levels which may be specified temporarily to resolve collisions in complex situations – see Secció “Real music example” in *Manual d’aprenentatge*.

Notes are only merged if they have opposing stem directions (as they have, for example, in voices one and two by default or when the stems are explicitly set in opposite directions).

Instruccions predefinides

`\mergeDifferentlyDottedOn`, `\mergeDifferentlyDottedOff`, `\mergeDifferentlyHeadedOn`, `\mergeDifferentlyHeadedOff`.

`\shiftOn`, `\shiftOnn`, `\shiftOnnn`, `\shiftOff`.

Fragments de codi seleccionats

Additional voices to avoid collisions

In some instances of complex polyphonic music, additional voices are necessary to prevent collisions between notes. If more than four parallel voices are needed, additional voices can be added by defining a variable using the 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
}
>>
}

```



Forcing horizontal shift of notes

When the typesetting engine cannot cope, the following syntax can be used to override typesetting decisions. The units of measure used here are staff spaces.

```

\relative c' <<
{
<d g>2 <d g>
}
\\
{
<b f'>2
\once \override NoteColumn.force-hshift = #1.7
<b f'>2
}
>>

```



Vegeu també

Music Glossary: Secció “polyphony” in *Glossari musical*.

Learning Manual: Secció “Multiple notes at once” in *Manual d’aprenentatge*, Secció “Voices contain music” in *Manual d’aprenentatge*, Secció “Real music example” in *Manual d’aprenentatge*.

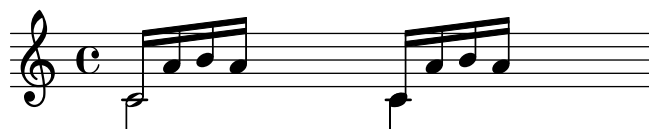
Snippets: Secció “Simultaneous notes” in *Fragments de codi*.

Internals Reference: Secció “NoteColumn” in *Referència de funcionament intern*, Secció “NoteCollision” in *Referència de funcionament intern*, Secció “RestCollision” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Using `\override NoteColumn.ignore-collision = ##t` will cause differently headed notes in different voices to merge incorrectly.

```
\mergeDifferentlyHeadedOn
<< \relative { c'16 a' b a } \\\ \relative { c'2 } >>
\override NoteColumn.ignore-collision = ##t
<< \relative { c'16 a' b a } \\\ \relative { c'2 } >>
```



Automatic part combining

Automatic part combining is used to merge two separate parts of music onto a single staff. This can be especially helpful when typesetting orchestral scores. A single **Voice** is printed while the two parts of music are the same, but in places where they differ, a second **Voice** is printed. Stem directions are set up & down accordingly while Solo and a *due* parts are also identified and marked appropriately.

The syntax for automatic part combining is:

```
\partcombine musicexpr1 musicexpr2
```

The following example demonstrates the basic functionality, putting parts on a single staff as polyphony and setting stem directions accordingly. The same variables are used for the independent parts and the combined staff.

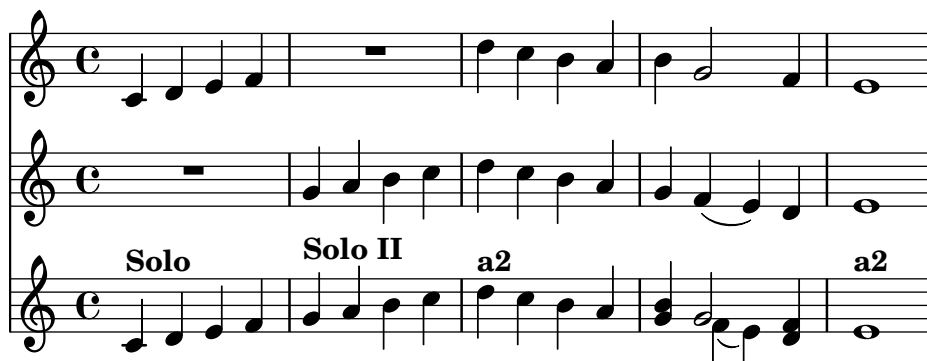
```
instrumentOne = \relative {
  c'4 d e f |
  R1 |
  d'4 c b a |
  b4 g2 f4 |
  e1 |
}

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
>>

```



Both parts have identical notes in the third measure, so only one instance of the notes is printed. Stem, slur, and tie directions are set automatically, depending on whether the parts are playing solo or in unison. When needed in polyphony situations, the first part (with context called *one*) gets “up” stems, while the second (called *two*) always gets “down” stems. In solo situations, the first and second parts get marked with “Solo” and “Solo II”, respectively. The unison (*a due*) parts are marked with the text “a2”.

By default, the partcombiner merges two notes of the same pitch as an *a due* note, combines notes with the same rhythm less than a ninth apart as chords and separates notes more than a ninth apart (or when the voices cross) into separate voices. This can be overridden with an optional argument of a pair of numbers after the `\partcombine` command: the first specifies the interval where notes start to be combined (the default is zero) and the second where the notes are split into separate voices. Setting the second argument to zero means that the partcombiner splits notes with an interval of a second or more, setting it to one splits notes of a third or more, and so on.

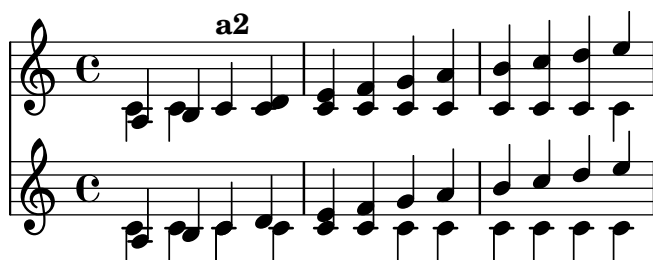
```

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
>>

```



Both arguments to `\partcombine` will be interpreted as separate `Voice` contexts, so if the music is being specified in relative mode then *both* parts must contain a `\relative` function, i.e.,

```
\partcombine
  \relative ... musicexpr1
  \relative ... musicexpr2
```

A `\relative` section that encloses a `\partcombine` has no effect on the pitches of *musicexpr1* or *musicexpr2*.

In professional scores, voices are often kept apart from each other for long passages of music even if some of the notes are the same in both voices, and could just as easily be printed as unison. Combining notes into a chord, or showing one voice as solo is, therefore, not ideal as the `\partcombine` function considers each note separately. In this case the `\partcombine` function can be overridden with one of the following commands. All of the commands may be preceded with `\once` in order to have them only apply to the next note in the music expression.

- `\partcombineApart` keeps the notes as two separate voices, even if they can be combined into a chord or unison.
- `\partcombineChords` combines the notes into a chord.
- `\partcombineUnisono` combines both voices as “unison”.
- `\partcombineSoloI` prints only voice one, and marks it as a “Solo”.
- `\partcombineSoloII` prints only voice two and marks it as a “Solo”.
- `\partcombineAutomatic` ends the functions of the commands above, and reverts back to the standard `\partcombine` functionality.

```
instrumentOne = \relative c' {
  \partcombineApart c2^"apart" e |
  \partcombineAutomatic e2^"auto" e |
  \partcombineChords e'2^"chord" e |
  \partcombineAutomatic c2^"auto" c |
  \partcombineApart c2^"apart" \once \partcombineChords e^"chord once" |
  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 in treble clef, each with a common time signature (C). The notation illustrates different ways to combine parts using the `\partcombine` command. Labels above the notes indicate the settings used: 'apart', 'auto', 'chord', 'auto', 'apart', and 'chord once'. The bottom staff also includes the label 'a2' for some notes, indicating a specific part combine setting.

Using `\partcombine` with lyrics

The `\partcombine` command is not designed to work with lyrics; if one of the voices is explicitly named in order to attach lyrics to it, the part combiner will stop working. However, this effect can be achieved using a `NullVoice` context. See [Polyphony with shared lyrics], pàgina 284.

Fragments de codi seleccionats

Combining two parts on the same staff

The part combiner tool (`\partcombine` command) allows the combination of several different parts on the same staff. Text directions such as “solo” or “a2” are added by default; to remove them, simply set the property `printPartCombineTexts` to `f`. For vocal scores (hymns), there is no need to add “solo/a2” texts, so they should be switched off. However, it might be better not to use it if there are any solos, as they won’t be indicated. In such cases, standard polyphonic notation may be preferable.

This snippet presents the three ways two parts can be printed on a same staff: standard polyphony, `\partcombine` without texts, and `\partcombine` with texts.

%% 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	

Changing partcombine texts

When using the automatic part combining feature, the printed text for the solo and unison sections may be changed:

```

\new Staff <<
  \set Staff.soloText = #"girl"
  \set Staff.soloIIText = #"boy"
  \set Staff.aDueText = #"together"
  \partcombine
    \relative c'' {
      g4 g r r
      a2 g
    }
    \relative c'' {
      r4 r a( b)
      a2 g
    }
  >>

```



Vegeu també

Music Glossary: Secció “a due” in *Glossari musical*, Secció “part” in *Glossari musical*.

Notation Reference: Secció 1.6.3 [Writing parts], pàgina 202.

Snippets: Secció “Simultaneous notes” in *Fragments de codi*.

Internals Reference: Secció “PartCombineMusic” in *Referència de funcionament intern*, Secció “Voice” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

All `\partcombine...` functions can only accept two voices.

`\partcombine...` functions cannot be placed inside a `\tuplet` or `\relative` block.

If `\printPartCombineTexts` is set and the two voices play the same notes “on and off”, in the same measure, the part combiner may typeset `a2` more than once in that measure.

`\partcombine` only knows when a note starts in a `Voice`; it cannot, for example, remember if a note in one `Voice` has already started when combining notes that have just started in the other `Voice`. This can lead to a number of unexpected issues including “Solo” or “Unison” marks being printed incorrectly.

`\partcombine` keeps all spanners (slurs, ties, hairpins, etc.) in the same `Voice` so that if any such spanners start or end in a different `Voice`, they may not be printed properly or at all.

If the `\partcombine` function cannot combine both music expressions (i.e., when both voices have different durations), it will give the voices, internally, its own custom names: `one` and `two` respectively. This means if there is any “switch” to a differently named `Voice` context, the events in that differently named `Voice` will be ignored.

Refer also to *Known issues and warnings* when using `\partcombine` with tablature in [Default tablatures], pàgina 334, and the *Note* in [Automatic beams], pàgina 82, when using automatic beaming.

Writing music in parallel

Music for multiple parts can be interleaved in input code. The function `\parallelMusic` accepts a list with the names of a number of variables to be created, and a musical expression. The content of alternate measures from the expression become the value of the respective variables, so you can use them afterwards to print the music.

Nota: Bar checks `|` must be used, and the measures must be of the same length.

```
\parallelMusic #'(voiceA voiceB voiceC) {
  % Bar 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 |

  % Bar 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 }
>>
```

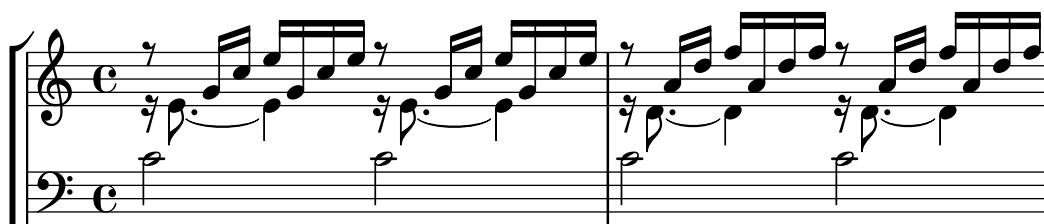


Relative mode may be used. Note that the `\relative` command is not used inside `\parallelMusic` itself. The notes are relative to the preceding note in the voice, not to the previous note in the input – in other words, relative notes for `voiceA` ignore the notes in `voiceB`.

```
\parallelMusic #'(voiceA voiceB voiceC) {
  % Bar 1
  r8 g16 c e g, c e r8 g,16 c e g, c e |
  r16 e8.~ 4          r16 e8.~ 4          |
  c2                  c                  |

  % Bar 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 }
>>
```



This works quite well for piano music. This example maps four consecutive measures to four variables:

```
global = {
  \key g \major
  \time 2/4
}

\parallelMusic #'(voiceA voiceB voiceC voiceD) {
  % Bar 1
  a8    b    c    d    |
  d4          e    |
  c16 d e fis d e fis g |
  a4          a    |
```

```

% Bar 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
      >>
    }
  >>
}

```



Vegeu també

Learning Manual: Secció “Organizing pieces with variables” in *Manual d’aprenentatge*.

Snippets: Secció “Simultaneous notes” in *Fragments de codi*.

1.6 Staff notation

Trumpet Bb

Tambourine

Piano

Comodo

p grazioso

This section explains how to influence the appearance of staves, how to print scores with more than one staff, and how to add tempo indications and cue notes to staves.

1.6.1 Displaying staves

This section describes the different methods of creating and grouping staves.

Instantiating new staves

Staves (singular: *staff*) are created with the `\new` or `\context` commands. For details, see Secció 5.1.2 [Creating and referencing contexts], pàgina 573.

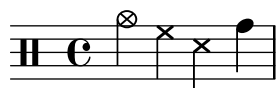
The basic staff context is `Staff`:

```
\new Staff \relative { c''4 d e f }
```



The `DrumStaff` context creates a five-line staff set up for a typical drum set. Each instrument is shown with a different symbol. The instruments are entered in drum mode following a `\drummode` command, with each instrument specified by name. For details, see [Percussion staves], pàgina 383.

```
\new DrumStaff {
  \drummode { cymc hh ss tomh }
}
```



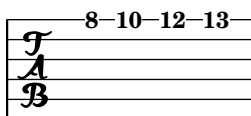
RhythmicStaff creates a single-line staff that only displays the rhythmic values of the input. Real durations are preserved. For details, see [Showing melody rhythms], pàgina 79.

```
\new RhythmicStaff { c4 d e f }
```



TabStaff creates a tablature with six strings in standard guitar tuning. For details, see [Default tablatures], pàgina 334.

```
\new TabStaff \relative { c''4 d e f }
```



There are two staff contexts specific for the notation of ancient music: **MensuralStaff** and **VaticanaStaff**. They are described in [Pre-defined contexts], pàgina 426.

The **GregorianTranscriptionStaff** context creates a staff to notate modern Gregorian chant. It does not show bar lines.

```
\new GregorianTranscriptionStaff \relative { c''4 d e f e d }
```



New single staff contexts may be defined. For details, see Secció 5.1.6 [Defining new contexts], pàgina 586.

Vegeu també

Music Glossary: Secció “staff” in *Glossari musical*, Secció “staves” in *Glossari musical*.

Notation Reference: Secció 5.1.2 [Creating and referencing contexts], pàgina 573, [Percussion staves], pàgina 383, [Showing melody rhythms], pàgina 79, [Default tablatures], pàgina 334, [Pre-defined contexts], pàgina 426, [Staff symbol], pàgina 194, [Gregorian chant contexts], pàgina 435, [Mensural contexts], pàgina 428, Secció 5.1.6 [Defining new contexts], pàgina 586.

Snippets: Secció “Staff notation” in *Fragments de codi*.

Internals Reference: Secció “Staff” in *Referència de funcionament intern*, Secció “DrumStaff” in *Referència de funcionament intern*, Secció “GregorianTranscriptionStaff” in *Referència de funcionament intern*, Secció “RhythmicStaff” in *Referència de funcionament intern*, Secció “TabStaff” in *Referència de funcionament intern*, Secció “MensuralStaff” in *Referència de funcionament intern*, Secció “VaticanaStaff” in *Referència de funcionament intern*, Secció “StaffSymbol” in *Referència de funcionament intern*.

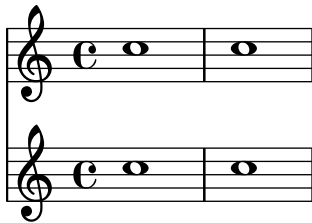
Grouping staves

Various contexts exist to group single staves together in order to form multi-stave systems. Each grouping context sets the style of the system start delimiter and the behavior of bar lines.

If no context is specified, the default properties will be used: the group is started with a vertical line, and the bar lines are not connected.

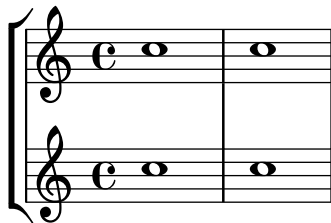
```
<<
\new Staff \relative { c''1 c }
\new Staff \relative { c''1 c }
```

>>



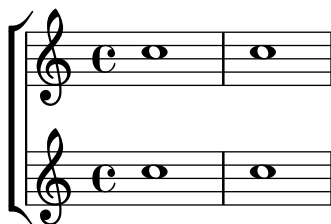
In the `StaffGroup` context, the group is started with a bracket and bar lines are drawn through all the staves.

```
\new StaffGroup <<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



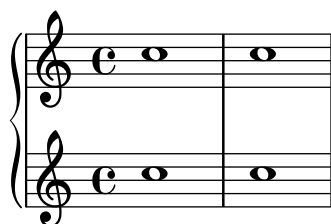
In a `ChoirStaff`, the group starts with a bracket, but bar lines are not connected.

```
\new ChoirStaff <<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



In a `GrandStaff`, the group begins with a brace, and bar lines are connected between the staves.

```
\new GrandStaff <<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



The `PianoStaff` is identical to a `GrandStaff`, except that it supports printing the instrument name directly. For details, see [Instrument names], pàgina 202.

```
\new PianoStaff <<
  \set PianoStaff.instrumentName = #"Piano"
  \new Staff \relative { c'1 c }
  \new Staff \relative { \clef bass c1 c }
>>
```



Each staff group context sets the property `systemStartDelimiter` to one of the following values: `SystemStartBar`, `SystemStartBrace`, or `SystemStartBracket`. A fourth delimiter, `SystemStartSquare`, is also available, but it must be explicitly specified.

New staff group contexts may be defined. For details, see Secció 5.1.6 [Defining new contexts], pàgina 586.

Fragments de codi seleccionats

Use square bracket at the start of a staff group

The system start delimiter `SystemStartSquare` can be used by setting it explicitly in a `StaffGroup` or `ChoirStaff` context.

```
\score {
  \new StaffGroup { <<
    \set StaffGroup.systemStartDelimiter = #'SystemStartSquare
    \new Staff { c'4 d' e' f' }
    \new Staff { c'4 d' e' f' }
  >> }
}
```



Display bracket with only one staff in a system

If there is only one staff in one of the staff types `ChoirStaff` or `StaffGroup`, by default the bracket and the starting bar line will not be displayed. This can be changed by overriding `collapse-height` to set its value to be less than the number of staff lines in the staff.

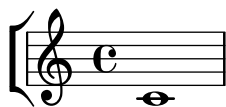
Note that in contexts such as `PianoStaff` and `GrandStaff` where the systems begin with a brace instead of a bracket, another property has to be set, as shown on the second system in the example.

```
\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
    }
  >>
}

```



Mensurstriche layout (bar lines between the staves)

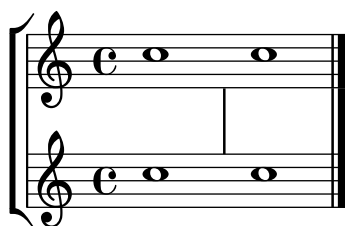
The mensurstriche-layout where the bar lines do not show on the staves but between staves can be achieved with a **StaffGroup** instead of a **ChoirStaff**. The bar line on staves is blanked out using `\hide`.

```

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 } >> }
  >>
}

```



Vegeu també

Music Glossary: Secció “brace” in *Glossari musical*, Secció “bracket” in *Glossari musical*, Secció “grand staff” in *Glossari musical*.

Notation Reference: [Instrument names], pàgina 202, Secció 5.1.6 [Defining new contexts], pàgina 586.

Snippets: Secció “Staff notation” in *Fragments de codi*.

Internals Reference: Secció “Staff” in *Referència de funcionament intern*, Secció “StaffGroup” in *Referència de funcionament intern*, Secció “ChoirStaff” in *Referència de funcionament intern*, Secció “GrandStaff” in *Referència de funcionament intern*, Secció “PianoStaff” in *Referència de funcionament intern*, Secció “SystemStartBar” in *Referència de funcionament intern*, Secció “SystemStartBrace” in *Referència de funcionament intern*, Secció “SystemStartBracket” in *Referència de funcionament intern*, Secció “SystemStartSquare” in *Referència de funcionament intern*.

Nested staff groups

Staff-group contexts can be nested to arbitrary depths. In this case, each child context creates a new bracket adjacent to the bracket of its parent group.

```
\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 }
    >>
  >>
>>
```



New nested staff group contexts can be defined. For details, see Secció 5.1.6 [Defining new contexts], pàgina 586.

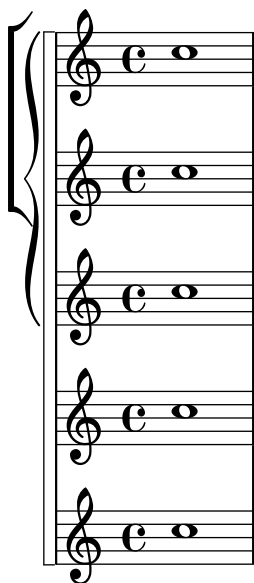
Fragments de codi seleccionats

Nesting staves

The property `systemStartDelimiterHierarchy` can be used to make more complex nested staff groups. The command `\set StaffGroup.systemStartDelimiterHierarchy` takes an alphabetical list of the number of staves produced. Before each staff a system start delimiter can be given. It has to be enclosed in brackets and takes as much staves as the brackets enclose. Elements in the list can be omitted, but the first bracket takes always the complete number of staves. The possibilities are `SystemStartBar`, `SystemStartBracket`, `SystemStartBrace`, and `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 }
>>
```



Vegeu també

Notation Reference: [Grouping staves], pàgina 187, [Instrument names], pàgina 202, Secció 5.1.6 [Defining new contexts], pàgina 586.

Snippets: Secció “Staff notation” in *Fragments de codi*.

Internals Reference: Secció “StaffGroup” in *Referència de funcionament intern*, Secció “ChoirStaff” in *Referència de funcionament intern*, Secció “SystemStartBar” in *Referència de funcionament intern*, Secció “SystemStartBrace” in *Referència de funcionament intern*, Secció “SystemStartBracket” in *Referència de funcionament intern*, Secció “SystemStartSquare” in *Referència de funcionament intern*.

Separating systems

If the number of systems per page changes from page to page it is customary to separate the systems by placing a system separator mark between them. By default the system separator is blank, but can be turned on with a `\paper` option.

```
\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
    % following commands are needed only to format this documentation
    paper-width = 100\mm
    paper-height = 100\mm
    tagline = ##f
  }
}
```



Vegeu també

Notation Reference: Secció 4.1 [Page layout], pàgina 521.

Snippets: Secció “Staff notation” in *Fragments de codi*.

1.6.2 Modifying single staves

This section explains how to change specific attributes of one staff: for example, modifying the number of staff lines or the staff size. Methods to start and stop staves and set ossia sections are also described.

Staff symbol

The `\stopStaff` and `\startStaff` commands can be used to stop or (re)start the staff lines respectively, from being printed at any point within a score.

```
\relative {
  \stopStaff f''4 d \startStaff g, e
  f'4 d \stopStaff g, e
  f'4 d \startStaff g, e
}
```



Instruccions predefinides

`\startStaff, \stopStaff.`

The lines of a staff belong to the `StaffSymbol` grob (including ledger lines) and can be modified using `StaffSymbol` properties, but these modifications must be made before the staff is (re)started.

The number of staff lines can be altered:

```
\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 |
}
```



The position of each staff line can also be altered. A list of numbers sets each line's position. 0 corresponds to the normal center line, and the normal line positions are (-4 -2 0 2 4). A single staff line is printed for every value entered so that the number of staff lines, as well as their position, can be changed with a single override.

```
\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 |
}
```



To preserve typical stem directions (in the bottom half of the staff stems point up, in the top half they point down), align the center line (or space) of the customized staff with the position of the normal center line (0). The clef position and the position of middle C may need to be adjusted accordingly to fit the new lines. See [\[Clef\]](#), pàgina [\[Clef\]](#).

Staff line thickness can be altered. Ledger lines and note stems, by default, are also affected.

```
\new Staff \with {
  \override StaffSymbol.thickness = #3
} \relative {
  f''4 d g, e
}
```



It is also possible to set ledger line thickness independently of staff lines.

```
\new Staff \with {
  \override StaffSymbol.thickness = #2
  \override StaffSymbol.ledger-line-thickness = #'(0.5 . 0.4)
} \relative {
  f'''4 a, a,, f
}
```



The first value is multiplied by the staff line thickness, the second by the staff space and then the two values are added together to give the new thickness of the ledger line.

The vertical positions of ledger lines can be altered,

```
\new Staff \with {
  \override StaffSymbol.ledger-positions = #'(-3 -2 -1 2 5 6)
} \relative {
  f'''4 a, a,, f
}
```



Additional ledger lines can be made to appear above or below note heads depending on the current position relative to other note heads that also have their own ledger lines.

```
\new Staff \with {
  \override StaffSymbol.ledger-extra = #4
} \relative {
  f'''4 a, d, f,
}
```

}



Ledger lines can also be made to appear inside the staff where custom staff lines are required. The example shows the default position of ledger lines when the explicit `ledger-position` is and is not set. The `\stopStaff` is needed in the example to revert the `\override` for the whole `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
}
```



The distance between staff lines can be altered. This affects ledger line spacing as well.

```
\new Staff \with {
  \override StaffSymbol.staff-space = #1.5
} \relative {
  f'''4 d, g, e,
}
```



Fragments de codi seleccionats

Making some staff lines thicker than the others

For educational purposes, a staff line can be thickened (e.g., the middle line, or to emphasize the line of the G clef). This can be achieved by adding extra lines very close to the line that should be emphasized, using the `line-positions` property of the `StaffSymbol` object.

```
{
  \override Staff.StaffSymbol.line-positions =
    #'(-4 -2 -0.2 0 0.2 2 4)
  d'4 e' f' g'
}
```



Vegeu també

Music Glossary: Secció “line” in *Glossari musical*, Secció “ledger line” in *Glossari musical*, Secció “staff” in *Glossari musical*.

Notation Reference: `\new Staff` [Clef], pàgina `\new Staff`.

Snippets: Secció “Staff notation” in *Fragments de codi*.

Internals Reference: Secció “StaffSymbol” in *Referència de funcionament intern*, Secció “staff-symbol-interface” in *Referència de funcionament intern*.

Ossia staves

Ossia staves can be set by creating a new simultaneous staff in the appropriate location:

```
\new Staff \relative {
  c' '4 b d c
  <<
    { c4 b d c }
    \new Staff { e4 d f e }
  >>
  c4 b c2
}
```

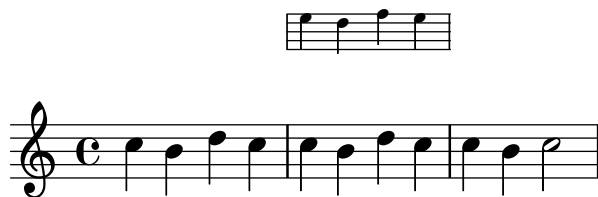


However, the above example is not what is usually desired. To create ossia staves that are above the original staff, have no time signature or clef, and have a smaller font size, tweaks must be used. The Learning Manual describes a specific technique to achieve this goal, beginning with Secció “Nesting music expressions” in *Manual d’aprenentatge*.

The following example uses the `alignAboveContext` property to align the ossia staff. This method is most appropriate when only a few ossia staves are needed.

```
\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
}
```



If many isolated ossia staves are needed, creating an empty **Staff** context with a specific *context id* may be more appropriate; the ossia staves may then be created by *calling* this context and using `\startStaff` and `\stopStaff` at the desired locations. The benefits of this method are more apparent if the piece is longer than the following example.

```
<<
  \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
  }
>>
```



Using the `\RemoveAllEmptyStaves` command to create ossia staves may be used as an alternative. This method is most convenient when ossia staves occur immediately following a line break. For more information about `\RemoveAllEmptyStaves`, see [Hiding staves], pàgina 200.


```

<<
\new Staff = "ossia" \with {
  \remove "Time_signature_engraver"
  \hide Clef
  \magnifyStaff #2/3
  \RemoveAllEmptyStaves
} \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
}
>>

```



Fragments de codi seleccionats

Vertically aligning ossias and lyrics

This snippet demonstrates the use of the context properties `alignBelowContext` and `alignAboveContext` to control the positioning of lyrics and ossias.

```

\paper {
  ragged-right = ##t
}

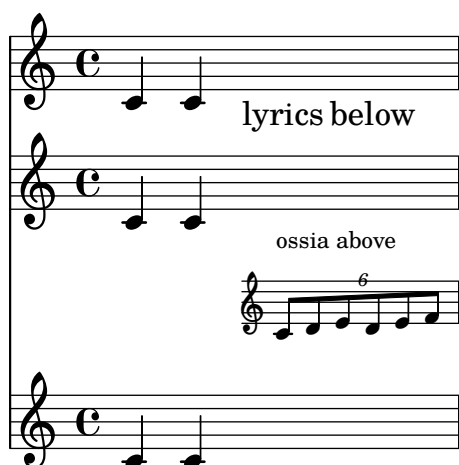
\relative c' <<
\new Staff = "1" { c4 c s2 }
\new Staff = "2" { c4 c s2 }
\new Staff = "3" { c4 c s2 }
{ \skip 2
  <<
    \lyrics {
      \set alignBelowContext = #"1"
      lyrics4 below
    }
    \new Staff \with {

```

```

        alignAboveContext = #"3"
        fontSize = #-2
        \override StaffSymbol.staff-space = #(magstep -2)
        \remove "Time_signature_engraver"
      } {
        \tuplet 6/4 {
          \override TextScript.padding = #3
          c8[^"ossia above" d e d e f]
        }
      }
    }
  }
  >>
}
>>

```



Vegeu també

Music Glossary: Secció “ossia” in *Glossari musical*, Secció “staff” in *Glossari musical*, Secció “Frenched staff” in *Glossari musical*.

Learning Manual: Secció “Nesting music expressions” in *Manual d’aprenentatge*, Secció “Size of objects” in *Manual d’aprenentatge*, Secció “Length and thickness of objects” in *Manual d’aprenentatge*.

Notation Reference: [Hiding staves], pàgina 200.

Snippets: Secció “Staff notation” in *Fragments de codi*.

Internals Reference: Secció “StaffSymbol” in *Referència de funcionament intern*.

Hiding staves

Staff lines can be hidden by removing the `Staff_symbol_engraver` from the `Staff` context. As an alternative, `\stopStaff` may be used.

```

\new Staff \with {
  \remove "Staff_symbol_engraver"
}
\relative { a''8 f e16 d c b a2 }

```



Empty staves can be hidden (for a so-called ‘Frenched Score’) by applying the `\RemoveEmptyStaves` command on a context, which can be done globally (in a `\layout` block) as well as for specific staves only (in a `\with` block). This command removes all empty staves in a score except for those in the first system. If you want those in the first system to be hidden also, use `\RemoveAllEmptyStaves`. Supported contexts are `Staff`, `RhythmicStaff` and `VaticanaStaff`.

Nota: A staff is considered empty when it contains only multi-measure rests, rests, skips, spacer rests, or a combination of these elements.

```
\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
  }
  >>
```



`\RemoveAllEmptyStaves` can also be used to create ossia sections for a staff. For details, see [Ossia staves], pàgina 197.

Instruccions predefinides

`\RemoveEmptyStaves`, `\RemoveAllEmptyStaves`.

Vegeu també

Music Glossary: Secció “Frenched staff” in *Glossari musical*.

Learning Manual: Secció “Visibility and color of objects” in *Manual d’aprenentatge*.

Notation Reference: Secció 5.1.5 [Changing context default settings], pàgina 581, [Staff symbol], pàgina 194, [Ossia staves], pàgina 197, [Hidden notes], pàgina 221, [Invisible rests], pàgina 59, Secció 5.4.7 [Visibility of objects], pàgina 610.

Snippets: Secció “Staff notation” in *Fragments de codi*.

Internals Reference: Secció “ChordNames” in *Referència de funcionament intern*, Secció “FiguredBass” in *Referència de funcionament intern*, Secció “Lyrics” in *Referència de funcionament intern*, Secció “Staff” in *Referència de funcionament intern*, Secció “VerticalAxisGroup” in *Referència de funcionament intern*, Secció “Staff_symbol_engraver” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Removing `Staff_symbol_engraver` also hides bar lines. If bar line visibility is forced, formatting errors may occur. In this case, use the following overrides instead of removing the engraver:

```
\omit StaffSymbol
\override NoteHead.no-ledgers = ##t
```

For the Known issues and warnings associated with `\Staff \RemoveEmptyStaves` see Secció 5.1.5 [Changing context default settings], pàgina 581.

1.6.3 Writing parts

This section explains how to insert tempo indications and instrument names into a score. Methods to quote other voices and format cue notes are also described.

Instrument names

Instrument names can be printed on the left side of staves in the `Staff`, `PianoStaff`, `StaffGroup`, `GrandStaff` and `ChoirStaff` contexts. The value of `instrumentName` is used for the first staff, and the value of `shortInstrumentName` is used for all succeeding staves.

```
\new Staff \with {
  instrumentName = #"Violin "
  shortInstrumentName = #"Vln. "
} \relative {
  c'4.. g'16 c4.. g'16 \break | c1 |
}
```



`\markup` can be used to create more complex instrument names:

```
\new Staff \with {
```

```

instrumentName = \markup {
  \column { "Clarinetti"
    \line { "in B" \smaller \flat }
  }
}
} \relative {
  c''4 c,16 d e f g2
}

```



When two or more staff contexts are grouped together, the instrument names and short instrument names are centered by default. To center multi-line instrument names, `\center-column` must be used:

```

<<
  \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 }
>>

```



However, if the instrument names are longer, the instrument names in a staff group may not be centered unless the `indent` and `short-indent` settings are increased. For details about these settings, see [\[\paper variables for shifts and indents\]](#), pàgina 528.

```

<<
  \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.

To add instrument names to other contexts (such as `ChordNames` or `FiguredBass`), `Instrument_name_engraver` must be added to that context. For details, see Secció 5.1.4 [Modifying context plug-ins], pàgina 579.

The `shortInstrumentName` may be changed in the middle of a piece, along with other settings as needed for the new instrument. However, only the first instance of `instrumentName` will be printed and subsequent changes will be ignored:

```

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'
}

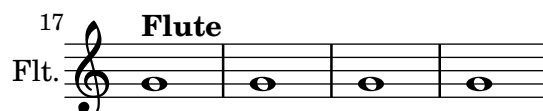
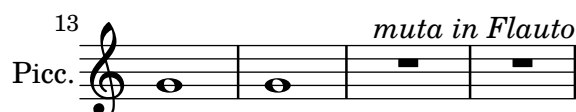
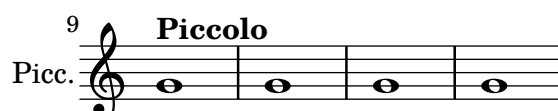
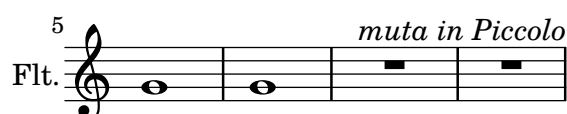
```

```

}

\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
}

```



Vegeu també

Notation Reference: [\paper variables for shifts and indents], pàgina 528, Secció 5.1.4 [Modifying context plug-ins], pàgina 579.

Snippets: Secció “Staff notation” in *Fragments de codi*.

Internals Reference: Secció “InstrumentName” in *Referència de funcionament intern*, Secció “PianoStaff” in *Referència de funcionament intern*, Secció “Staff” in *Referència de funcionament intern*.

Quoting other voices

It is very common for one voice to use the same notes as those from another voice. For example, first and second violins playing the same phrase during a particular passage of the music. This is done by letting one voice *quote* the other, without having to re-enter the music all over again for the second voice.

The `\addQuote` command, used in the top level scope, defines a stream of music from which fragments can be quoted.

The `\quoteDuring` command is used to indicate the point where the quotation begins. It is followed by two arguments: the name of the quoted voice, as defined with `\addQuote`, and a music expression for the duration of the quote.

```
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
  >>
}
```



If the music expression used in `\quoteDuring` contains notes instead of spacer or multimeasure rests then the quote will appear as polyphony and may produce unexpected results.

```
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
  >>
}

```



If an `\unfoldRepeat` command in a music expression is required to be printed when using `\quoteDuring`, then it too must also contain its own `\unfoldRepeat` command;

```

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)

The `\quoteDuring` command uses the `\transposition` settings of both quoted and quoting parts to produce notes for the quoting part that have the same sounding pitch as those in the quoted part.

```
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

By default quoted music will include all articulations, dynamics, markups, etc., in the quoted expression. It is possible to choose which of these objects from the quoted music are displayed by using the `quotedEventTypes` context property.

```
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
  >>
}

```



Quotes can also be tagged, see [Using tags], pàgina 498.

Vegeu també

Notation Reference: [\(undefined\)](#) [Instrument transpositions], pàgina [\(undefined\)](#), [Using tags], pàgina 498.

Installed Files: `scm/define-event-classes.scm`.

Snippets: Secció “Staff notation” in *Fragments de codi*.

Internals Reference: Secció “Music classes” in *Referència de funcionament intern*, Secció “QuoteMusic” in *Referència de funcionament intern*, Secció “Voice” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Only the contents of the first **Voice** occurring in an `\addQuote` command will be considered for quotation, so if the music expression contains `\new` or `\context Voice` statements, their contents will not be quoted. Quoting grace notes is unsupported and may cause LilyPond to crash whereas quoting nested triplets may result in poor notation.

Formatting cue notes

The simplest way to format cue notes is to explicitly create a **CueVoice** context within the part.

```

\relative {
  R1
  <<
    { e'2\rest r4. e8 }
    \new CueVoice {
      \stemUp d'8~"flute" c d e fis2
    }
  >>
  d,4 r a r
}

```



The `\cueClef` command can also be used with an explicit `CueVoice` context if a change of clef is required and will print an appropriately sized clef for the cue notes. The `\cueClefUnset` command can then be used to switch back to the original clef, again with an appropriately sized clef.

```
\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
}
```



The `\cueClef` and `\cueClefUnset` command can also be used without a `CueVoice` if required.

```
\relative {
  \clef "bass"
  R1
  \cueClef "treble"
  d''8^"flute" c d e fis2
  \cueClefUnset
  d,,4 r a r
}
```



For more complex cue note placement, e.g including transposition, or inserting cue notes from multiple music sources the `\cueDuring` or `\cueDuringWithClef` commands can be used. These are more specialized form of `\quoteDuring`, see [Quoting other voices], pàgina 206, in the previous section.

The syntax is:

```
\cueDuring #quotename #direction #music
and
\cueDuringWithClef #quotename #direction #clef #music
```

The music from the corresponding measures of the *quote name* is added as a `CueVoice` context and occurs simultaneously with the *music*, which then creates a polyphonic situation. The *direction* takes the argument UP or DOWN, and corresponds to the first and second voices respectively, determining how the cue notes are printed in relation to the other voice.

```
fluteNotes = \relative {
  r2. c''4 | d8 c d e fis2 | g2 d |
}
```

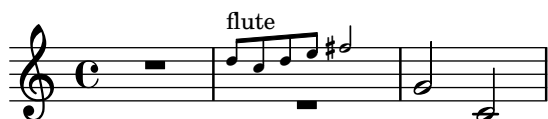
```

oboeNotes = \relative c'' {
  R1
  <>^\markup \tiny { flute }
  \cueDuring #"flute" #UP { R1 }
  g2 c,
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \oboeNotes
}

```



It is possible to adjust which aspects of the music are quoted with `\cueDuring` by setting the `quotedCueEventTypes` property. Its default value is `'(note-event rest-event tie-event beam-event tuplet-span-event)`, which means that only notes, rests, ties, beams and tuplets are quoted, but not articulations, dynamic marks, markup, etc.

Nota: When a Voice starts with `\cueDuring`, as in the following example, the Voice context must be explicitly declared, or else the entire music expression would belong to the `CueVoice` context.

```

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,
}

```



Markup can be used to show the name of the quoted instrument. If the cue notes require a change in clef, this can be done manually but the original clef should also be restored manually at the end of the cue notes.

```

fluteNotes = \relative {
  r2. c''4 d8 c d e fis2 g2 d2
}

```

```

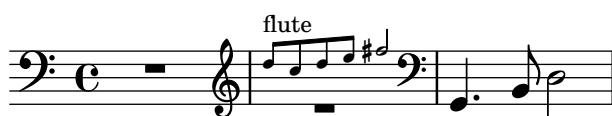
}

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
}

```



Alternatively, the `\cueDuringWithClef` function can be used instead. This command takes an extra argument to specify the change of clef that needs to be printed for the cue notes but will automatically print the original clef once the cue notes have finished.

```

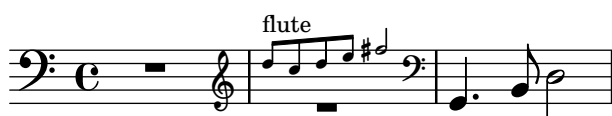
fluteNotes = \relative {
  r2. c''4 d8 c d e fis2 g2 d2
}

bassoonNotes = \relative c {
  \clef bass
  R1
  <>^\markup { \tiny "flute" }
  \cueDuringWithClef #"flute" #UP #"treble" { R1 }
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}

```



Like `\quoteDuring`, `\cueDuring` takes instrument transpositions into account. Cue notes are produced at the pitches that would be written for the instrument receiving the cue to produce the sounding pitches of the source instrument.

To transpose cue notes differently, use `\transposedCueDuring`. This command takes an extra argument to specify (in absolute mode) the printed pitch that you want to represent the sound

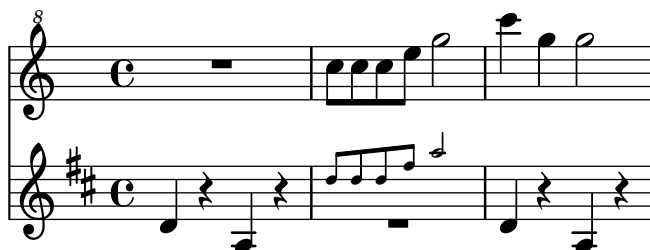
of a concert middle C. This is useful for taking cues from an instrument in a completely different register.

```
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
>>
```



The `\killCues` command removes cue notes from a music expression, so the same music expression can be used to produce the instrument part with cues and the score. The `\killCues` command removes only the notes and events that were quoted by `\cueDuring`. Other markup associated with cues, such as clef changes and a label identifying the source instrument, can be tagged for selective inclusion in the score; see [Using tags], pàgina 498.

```
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
}
```

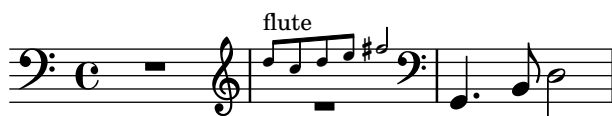
```

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}

\new StaffGroup <<
  \new Staff {
    \fluteNotes
  }
  \new Staff {
    \removeWithTag #'part { \killCues { \bassoonNotes } }
  }
>>

```



Alternatively, Clef changes and instrument labels can be collected into an instrument definition for repeated use, using `\addInstrumentDefinition` described in [Instrument names], pàgina 202.

Vegeu també

Notation Reference: [Quoting other voices], pàgina 206, <undefined> [Instrument transpositions], pàgina <undefined>, [Instrument names], pàgina 202, <undefined> [Clef], pàgina <undefined>, [Musical cues], pàgina 300, [Using tags], pàgina 498.

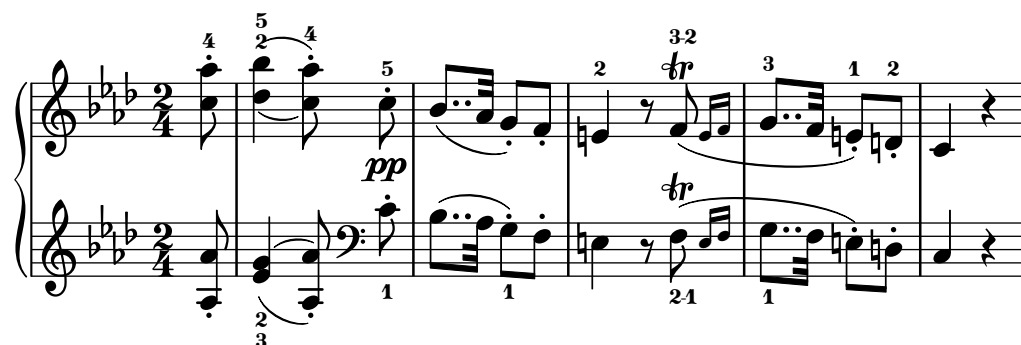
Snippets: Secció “Staff notation” in *Fragments de codi*.

Internals Reference: Secció “CueVoice” in *Referència de funcionament intern*, Secció “Voice” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Collisions can occur with rests, when using `\cueDuring`, between `Voice` and `CueVoice` contexts. When using `\cueDuringWithClef` or `\transposedCueDuring` the extra argument required for each case must come after the quote and the direction.

1.7 Editorial annotations



This section discusses the various ways to change the appearance of notes and add analysis or educational emphasis.

1.7.1 Inside the staff

This section discusses how to add emphasis to elements that are inside the staff.

Selecting notation font size

Nota:

For font sizes of text, see [Selecting font and font size], pàgina 239.

For staff size, see Secció 4.2.2 [Setting the staff size], pàgina 533.

For cue notes, see [Formatting cue notes], pàgina 209.

For ossia staves, see [Ossia staves], pàgina 197.

To change the size of the notation without changing the staff size, specify a magnification factor with the `\magnifyMusic` command:

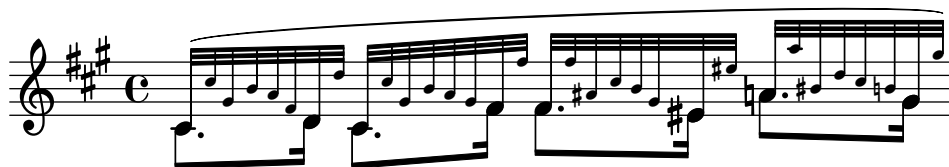
```
\new Staff <<
  \new Voice \relative {
    \voiceOne
    <e' e'>4 <f f'>8. <g g'>16 <f f'>8 <e e'>4 r8
  }
  \new Voice \relative {
    \voiceTwo
    \magnifyMusic 0.63 {
      \override Score.SpacingSpanner.spacing-increment = #(* 1.2 0.63)
      r32 c' a c a c a c r c a c a c a c
      r c a c a c a c a c a c a c a c
    }
  }
}>>
```



The `\override` in the example above is a bug workaround. See the “Known issues and warnings” at the end of this section.

If a normal sized note head is merged with a smaller one, the size of the smaller note may need to be reset (with ‘\once \normalsize’) so that the stems and accidentals align properly:

```
\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
  }
}>>
```



The `\magnifyMusic` command is not intended for cue notes, grace notes, or ossia staves—there are more appropriate methods of entering each of those constructs. Instead, it is useful when the notation size changes in a single instrumental part on one staff, and where grace notes are not appropriate, such as in cadenza-like passages or in cases such as the above examples. Setting the `\magnifyMusic` value to 0.63 duplicates the dimensions of the `CueVoice` context.

Nota: The `\magnifyMusic` command should *not* be used when also resizing the staff. See Secció 4.2.2 [Setting the staff size], pàgina 533.

Resizing individual layout objects

An individual layout object can be resized by using the `\tweak` or `\override` commands to adjust its `font-size` property:

```
\relative {
  % resize a note head
  <f' \tweak font-size -4 b e>-5
  % resize a fingering
  bes-\tweak font-size 0 -3
  % resize an accidental
  \once \override Accidental.font-size = -4 bes!-^
  % resize an articulation
  \once \override Script.font-size = 4 bes!-^
}
```



The default `font-size` value for each layout object is listed in the Internals Reference. The `font-size` property can only be set for layout objects that support the `font-interface` layout interface. If `font-size` is not specified in the object's 'Standard settings' list, its value is 0. See Secció "All layout objects" in *Referència de funcionament intern*.

Understanding the `fontSize` property

The `fontSize` context property adjusts the relative size of all glyph-based notational elements in a context:

```
\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.
}
```



The `fontSize` value is a number indicating the size relative to the standard size for the current staff height. The default `fontSize` is 0; adding 6 to any `fontSize` value doubles the printed size of the glyphs, and subtracting 6 halves the size. Each step increases the size by approximately 12%.

The scheme function `magnification->font-size` is provided for convenience since the logarithmic units of the `font-size` property are not entirely intuitive. For example, to adjust the musical notation to 75% of the default size, use:

```
\set fontSize = #(magnification->font-size 0.75)
```

The scheme function `magstep` does the opposite: it converts a `font-size` value into a magnification factor.

The `fontSize` property will only affect notational elements that are drawn with glyphs, such as noteheads, accidentals, scripts, etc. It will not affect the size of the staff itself, nor will it scale stems, beams, or horizontal spacing. To scale stems, beams, and horizontal spacing along with the notation size (without changing the staff size), use the `\magnifyMusic` command discussed above. To scale everything, including the staff size, see Secció 4.2.2 [Setting the staff size], pàgina 533.

Whenever the `fontSize context property` is set, its value is added to the value of the `font-size grob property` for individual layout objects, before any glyphs are printed. This can cause confusion when setting individual `font-size` properties while `fontSize` is already set:

```
% the default font-size for NoteHead is 0
% the default font-size for Fingering is -5
c''4-3

\set fontSize = -3
% the effective font size for NoteHead is now -3
% the effective font size for Fingering is now -8
```

```
c''4-3
```

```
\override Fingering.font-size = 0
% the effective font size for Fingering is now -3
c''4-3
```



The following shorthand commands are also available:

Command	Equivalent to	Relative size
<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%

```
\relative c'' {
  \teeny
  c4.-> d8---3
  \tiny
  c4.-> d8---3
  \small
  c4.-> d8---3
  \normalsize
  c4.-> d8---3
  \large
  c4.-> d8---3
  \huge
  c4.-> d8---3
}
```



Font size changes are achieved by scaling the design size that is closest to the desired size. The standard font size (for `font-size = 0`) depends on the standard staff height. For a 20pt staff, an 11pt font is selected.

Instruccions predefinides

`\magnifyMusic`, `\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`.

Vegeu també

Notation Reference: [Selecting font and font size], pàgina 239, Secció 4.2.2 [Setting the staff size], pàgina 533, [Formatting cue notes], pàgina 209, [Ossia staves], pàgina 197.

Installed Files: `ly/music-functions-init.ly`, `ly/property-init.ly`.

Snippets: Secció “Editorial annotations” in *Fragments de codi*.

Internals Reference: Secció “font-interface” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

There are currently two bugs that are preventing proper horizontal spacing when using `\magnifyMusic`. There is only one available workaround, and it is not guaranteed to work in every case. In the example below, replace the *mag* variable with your own value. You may also try removing one or both of the `\newSpacingSection` commands, and/or the `\override` and `\revert` commands:

```
\magnifyMusic mag {
  \newSpacingSection
  \override Score.SpacingSpanner.spacing-increment = #(* 1.2 mag)
  [music]
  \newSpacingSection
  \revert Score.SpacingSpanner.spacing-increment
}
```

Fingering instructions

Fingering instructions can be entered using *'note-digit'*:

```
\relative { c''4-1 d-2 f-4 e-3 }
```



Markup texts or strings may be used for finger changes.

```
\relative {
  c''4-1 d-2 f\finger \markup \tied-lyric #"4~3" c\finger "2 - 3"
}
```



A thumb-script can be added (e.g., cello music) to indicate that a note should be played with the thumb.

```
\relative { <a'_\thumb a'-3>2 <b'_\thumb b'-3> }
```



Fingerings for chords can also be added to individual notes by adding them after the pitches.

```
\relative {
  <c''-1 e-2 g-3 b-5>2 <d-1 f-2 a-3 c-5>
}
```



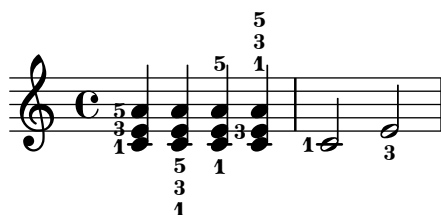
Fingering instructions may be manually placed above or below the staff, see Secció 5.4.2 [Direction and placement], pàgina 602.

Fragments de codi seleccionats

Controlling the placement of chord fingerings

The placement of fingering numbers can be controlled precisely. For fingering orientation to apply, you must use a chord construct `<>` even if it is a single note.

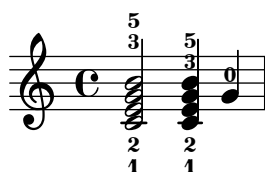
```
\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down right up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(left)
  <c-1>2
  \set fingeringOrientations = #'(down)
  <e-3>2
}
```



Allowing fingerings to be printed inside the staff

By default, vertically oriented fingerings are positioned outside the staff. However, this behavior can be canceled. Note: you must use a chord construct `<>`, even if it is only a single note.

```
\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>
}
```



Avoiding collisions with chord fingerings

Fingerings and string numbers applied to individual notes will automatically avoid beams and stems, but this is not true by default for fingerings and string numbers applied to the individual notes of chords. The following example shows how this default behavior can be overridden.

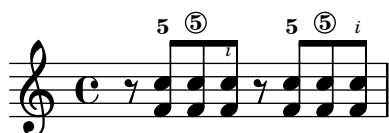
```
\relative c' {
  \set fingeringOrientations = #'(up)
  \set stringNumberOrientations = #'(up)
  \set strokeFingerOrientations = #'(up)
}
```

```

% 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
}

```



Vegeu també

Notation Reference: Secció 5.4.2 [Direction and placement], pàgina 602.

Snippets: Secció “Editorial annotations” in *Fragments de codi*.

Internals Reference: Secció “FingeringEvent” in *Referència de funcionament intern*, Secció “fingering-event” in *Referència de funcionament intern*, Secció “Fingering-engraver” in *Referència de funcionament intern*, Secció “New_fingering-engraver” in *Referència de funcionament intern*, Secció “Fingering” in *Referència de funcionament intern*.

Hidden notes

Hidden (or invisible or transparent) notes can be useful in preparing theory or composition exercises.

```

\relative {
  c''4 d
  \hideNotes
  e4 f
  \unHideNotes
  g a
  \hideNotes
  b
  \unHideNotes
  c
}

```



Note heads, stems, and flags, and rests are invisible. Beams are invisible if they start on a hidden note. Objects that are attached to invisible notes are still visible.

```

\relative c'' {

```

```
e8(\p f g a)--
\hideNotes
e8(\p f g a)--
}
```



Instruccions predefinides

`\hideNotes`, `\unHideNotes`.

Vegeu també

Learning Manual: Secció “Visibility and color of objects” in *Manual d’aprenentatge*.

Notation Reference: [Invisible rests], pàgina 59, Secció 5.4.7 [Visibility of objects], pàgina 610, [Hiding staves], pàgina 200.

Snippets: Secció “Editorial annotations” in *Fragments de codi*.

Internals Reference: Secció “Note-spacing-engraver” in *Referència de funcionament intern*, Secció “NoteSpacing” in *Referència de funcionament intern*.

Coloring objects

Individual objects may be assigned colors. Valid color names are listed in the Secció A.7 [List of colors], pàgina 649.

```
\override NoteHead.color = #red
c''4 c''
\override NoteHead.color = #(x11-color 'LimeGreen)
d''
\override Stem.color = #blue
e''
```



The full range of colors defined for X11 can be accessed by using the Scheme function `x11-color`. The function takes one argument; this can be a symbol in the form `'FooBar` or a string in the form `"FooBar"`. The first form is quicker to write and is more efficient. However, using the second form it is possible to access X11 colors by the multi-word form of its name.

If `x11-color` cannot make sense of the parameter then the color returned defaults to black.

```
\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)
```



```

c2 \parenthesize <c e g>
c2 <c \parenthesize e g>
}

```



Non-note objects may be parenthesized as well. For articulations, a hyphen is needed before the `\parenthesize` command.

```

\relative {
  c''2-\parenthesize -. d
  c2 \parenthesize r
}

```



Vegeu també

Snippets: Secció “Editorial annotations” in *Fragments de codi*.

Internals Reference: Secció “Parenthesis_engraver” in *Referència de funcionament intern*, Secció “ParenthesesItem” in *Referència de funcionament intern*, Secció “parentheses-interface” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Parenthesizing a chord prints parentheses around each individual note, instead of a single large parenthesis around the entire chord.

Stems

Whenever a note is found, a **Stem** object is created automatically. For whole notes and rests, they are also created but made invisible.

Stems may be manually placed to point up or down; see Secció 5.4.2 [Direction and placement], pàgina 602.

Instruccions predefinides

`\stemUp`, `\stemDown`, `\stemNeutral`.

Fragments de codi seleccionats

Default direction of stems on the center line of the staff

The default direction of stems on the center line of the staff is set by the **Stem** property `neutral-direction`.

```

\relative c'' {
  a4 b c b
  \override Stem.neutral-direction = #up
  a4 b c b
  \override Stem.neutral-direction = #down
  a4 b c b
}

```



Automatically changing the stem direction of the middle note based on the melody

LilyPond can alter the stem direction of the middle note on a staff so that it follows the melody, by adding the `Melody_engraver` to the Voice context and overriding the `neutral-direction` of 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 = #'()
  }
}
```



Vegeu també

Notation Reference: Secció 5.4.2 [Direction and placement], pàgina 602.

Snippets: Secció “Editorial annotations” in *Fragments de codi*.

Internals Reference: Secció “Stem_engraver” in *Referència de funcionament intern*, Secció “Stem” in *Referència de funcionament intern*, Secció “stem-interface” in *Referència de funcionament intern*.

1.7.2 Outside the staff

This section discusses how to add emphasis to elements in the staff from outside of the staff.

Balloon help

Elements of notation can be marked and named with the help of a square balloon. The primary purpose of this feature is to explain notation.

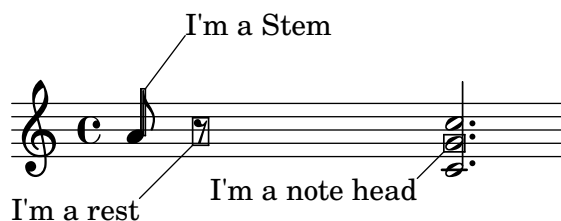
```
\new Voice \with { \consists "Balloon_engraver" }
\relative c'' {
  \balloonGrobText #'Stem #'(3 . 4) \markup { "I'm a Stem" }
  a8
  \balloonGrobText #'Rest #'(-4 . -4) \markup { "I'm a rest" }
  r
  <c, g'-\balloonText #'(-2 . -2) \markup { "I'm a note head" } c>2.
}
```



There are two music functions, `balloonGrobText` and `balloonText`; the former is used like `\once \override` to attach text to any grob, and the latter is used like `\tweak`, typically within chords, to attach text to an individual note.

Balloon text does not influence note spacing, but this can be altered:

```
\new Voice \with { \consists "Balloon_engraver" }
\relative c'' {
  \balloonGrobText #'Stem #'(3 . 4) \markup { "I'm a Stem" }
  a8
  \balloonGrobText #'Rest #'(-4 . -4) \markup { "I'm a rest" }
  r
  \balloonLengthOn
  <c, g'-\balloonText #'(-2 . -2) \markup { "I'm a note head" } c>2.
}
```



Instruccions predefinides

`\balloonLengthOn`, `\balloonLengthOff`.

Vegeu també

Snippets: Secció “Editorial annotations” in *Fragments de codi*.

Internals Reference: Secció “Balloon_engraver” in *Referència de funcionament intern*, Secció “BalloonTextItem” in *Referència de funcionament intern*, Secció “balloon-interface” in *Referència de funcionament intern*.

Grid lines

Vertical lines can be drawn between staves synchronized with the notes.

The `Grid_point_engraver` must be used to create the end points of the lines, while the `Grid_line_span_engraver` must be used to actually draw the lines. By default this centers grid lines horizontally below and to the left side of each note head. Grid lines extend from the middle lines of each staff. The `gridInterval` must specify the duration between the grid lines.

```
\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
      }
    >>
  }

```



Fragments de codi seleccionats

Grid lines: changing their appearance

The appearance of grid lines can be changed by overriding some of their properties.

```

\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
    }
  }
}

```



Vegeu també

Snippets: Secció “Editorial annotations” in *Fragments de codi*.

Internals Reference: Secció “Grid_line_span_engraver” in *Referència de funcionament intern*, Secció “Grid_point_engraver” in *Referència de funcionament intern*, Secció “GridLine” in *Referència de funcionament intern*, Secció “GridPoint” in *Referència de funcionament intern*, Secció “grid-line-interface” in *Referència de funcionament intern*, Secció “grid-point-interface” in *Referència de funcionament intern*.

Analysis brackets

Brackets are used in musical analysis to indicate structure in musical pieces. Simple horizontal brackets are supported.

```

\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative {
  c''2\startGroup
  d\stopGroup
}

```



Analysis brackets may be nested.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}
\relative {
  c' '4\startGroup\startGroup
  d4\stopGroup
  e4\startGroup
  d4\stopGroup\stopGroup
}
```



Vegeu també

Snippets: Secció “Editorial annotations” in *Fragments de codi*.

Internals Reference: Secció “Horizontal_bracket_engraver” in *Referència de funcionament intern*, Secció “HorizontalBracket” in *Referència de funcionament intern*, Secció “horizontal-bracket-interface” in *Referència de funcionament intern*, Secció “Staff” in *Referència de funcionament intern*.

1.8 Text

p con amabilità

ten.

tranqu. dolce

ten.

ten.

cantabile, con intimissimo sentimento, ma sempre molto dolce e semplice

non staccato

*molto **p**, sempre tranquillo ed egualmente, non rubato*

Red. *Red.*

Red. *Red.*

This section explains how to include text (with various formatting) in music scores. Some text elements that are not dealt with here are discussed in other specific sections: Secció 2.1 [Vocal music], pàgina 255, Secció 3.2 [Titles and headers], pàgina 470.

1.8.1 Writing text

This section introduces different ways of adding text to a score.

Nota: To write accented and special text (such as characters from other languages), simply insert the characters directly into the LilyPond file. The file must be saved as UTF-8. For more information, see [Text encoding], pàgina 502.

Text scripts

Simple “quoted text” indications may be added to a score, as demonstrated in the following example. Such indications may be manually placed above or below the staff, using the syntax described in Secció 5.4.2 [Direction and placement], pàgina 602.

```
\relative { a'8^"pizz." g f e a4-"scherz." f }
```

pizz.

scherz.

This syntax is actually a shorthand; more complex text formatting may be added to a note by explicitly using a `\markup` block, as described in Secció 1.8.2 [Formatting text], pàgina 237.

```
\relative {
  a'8^\markup { \italic pizz. } g f e
  a4_\markup { \tiny scherz. \bold molto } f }
```

pizz.

scherz. molto

By default, text indications do not influence the note spacing. However, their widths can be taken into account: in the following example, the first text string does not affect spacing, whereas the second one does.

```
\relative {
  a'8~"pizz." g f e
  \textLengthOn
  a4_"scherzando" f
}
```



In addition to text scripts, articulations can be attached to notes. For more information, see [Articulations and ornamentations], pàgina 119.

For more information about the relative ordering of text scripts and articulations, see Secció “Placement of objects” in *Manual d’aprenentatge*.

Instruccions predefinides

`\textLengthOn`, `\textLengthOff`.

Vegeu també

Learning Manual: Secció “Placement of objects” in *Manual d’aprenentatge*.

Notation Reference: Secció 1.8.2 [Formatting text], pàgina 237, Secció 5.4.2 [Direction and placement], pàgina 602, [Articulations and ornamentations], pàgina 119.

Snippets: Secció “Text” in *Fragments de codi*.

Internals Reference: Secció “TextScript” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Checking to make sure that text scripts and lyrics are within the margins requires additional calculations. In cases where slightly faster performance is desired, use

```
\override Score.PaperColumn.keep-inside-line = ##f
```

Text spanners

Some performance indications, e.g., *rallentando* or *accelerando*, are written as text and are extended over multiple notes with dotted lines. Such objects, called “spanners”, may be created from one note to another using the following syntax:

```
\relative {
  \override TextSpanner.bound-details.left.text = "rit."
  b'1\startTextSpan
  e,\stopTextSpan
}
```



The string to be printed is set through object properties. By default it is printed in italic characters, but different formatting can be obtained using `\markup` blocks, as described in Secció 1.8.2 [Formatting text], pàgina 237.

```
\relative {
```

```
\override TextSpanner.bound-details.left.text =
  \markup { \upright "rit." }
b'1\startTextSpan c
e,\stopTextSpan
}
```



The line style, as well as the text string, can be defined as an object property. This syntax is described in Secció 5.4.8 [Line styles], pàgina 616.

Instruccions predefinides

`\textSpannerUp`, `\textSpannerDown`, `\textSpannerNeutral`.

Advertiments i problemes coneguts

LilyPond is only able to handle one text spanner per voice.

Fragments de codi seleccionats

Dynamics text spanner postfix

Custom text spanners can be defined and used with hairpin and text crescendos. `\<` and `\>` produce hairpins by default, `\cresc` etc. produce text spanners by default.

% 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\decresc c4\!
}
```



Dynamics custom text spanner postfix

Postfix functions for custom crescendo text spanners. The spanners should start on the first note of the measure. One has to use `-\mycresc`, otherwise the spanner start will rather be assigned to the next note.

% Two functions for (de)crescendo spanners where you can explicitly give the
% spanner text.

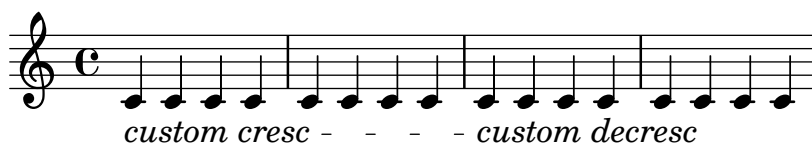
```
mycresc =
#(define-music-function (mymarkup) (markup?)
```

```

      (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 decresc" c4 c4 c4 |
  c4 c4\! c4 c4
}

```



Vegeu també

Notation Reference: Secció 5.4.8 [Line styles], pàgina 616, [Dynamics], pàgina 122, Secció 1.8.2 [Formatting text], pàgina 237.

Snippets: Secció “Text” in *Fragments de codi*, Secció “Expressive marks” in *Fragments de codi*.

Internals Reference: Secció “TextSpanner” in *Referència de funcionament intern*.

Text marks

Various text elements may be added to a score using the syntax described in [Rehearsal marks], pàgina 110:

```

\relative {
  \mark "Verse"
  c'2 g'
  \bar "||"
  \mark "Chorus"
  g2 c,
  \bar "|."
}

```



This syntax makes it possible to put any text on a bar line; more complex text formatting may be added using a `\markup` block, as described in Secció 1.8.2 [Formatting text], pàgina 237:

```

\relative {

```

```

<c' e>1
\mark \markup { \italic { colla parte } }
<d f>2 <e g>
<c f aes>1
}

```



This syntax also allows to print special signs, like coda, segno or fermata, by specifying the appropriate symbol name as explained in [Music notation inside markup], pàgina 248:

```

\relative {
  <bes' f>2 <aes d>
  \mark \markup { \musicglyph #"scripts.ufermata" }
  <e g>1
}

```



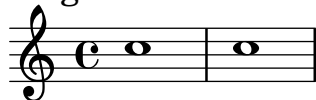
Such objects are only typeset above the top staff of the score; depending on whether they are specified at the end or the middle of a bar, they can be placed above the bar line or between notes. When specified at a line break, the mark will be printed at the beginning of the next line.

```

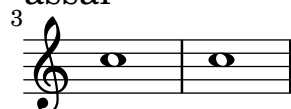
\relative c'' {
  \mark "Allegro"
  c1 c
  \mark "assai" \break
  c c
}

```

Allegro



assai



Instruccions predefinides

\markLengthOn, \markLengthOff.

Fragments de codi seleccionats

Printing marks at the end of a line

Marks can be printed at the end of the current line, instead of the beginning of the following line. In such cases, it might be preferable to align the right end of the mark with the bar line.

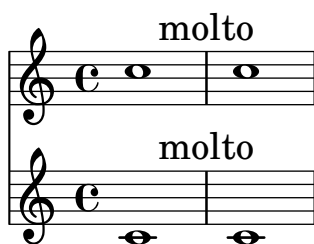
```
\relative c'' {
  g2 c
  d,2 a'
  \once \override Score.RehearsalMark.break-visibility = #end-of-line-visible
  \once \override Score.RehearsalMark.self-alignment-X = #RIGHT
  \mark "D.C. al Fine"
  \break
  g2 b,
  c1 \bar "||"
}
```



Printing marks on every staff

Although text marks are normally only printed above the topmost staff, they may also be printed on every staff.

```
\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"
    }
  }
}
```



Vegeu també

Notation Reference: [Rehearsal marks], pàgina 110, Secció 1.8.2 [Formatting text], pàgina 237, [Music notation inside markup], pàgina 248, Secció A.8 [The Feta font], pàgina 651.

Snippets: Secció “Text” in *Fragments de codi*.

Internals Reference: Secció “MarkEvent” in *Referència de funcionament intern*, Secció “Mark_engraver” in *Referència de funcionament intern*, Secció “RehearsalMark” in *Referència de funcionament intern*.

Separate text

A `\markup` block can exist by itself, outside of any `\score` block, as a “top-level expression”. This syntax is described in Secció 3.1.5 [File structure], pàgina 468.

```
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
```

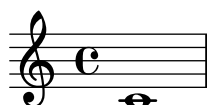
Tomorrow, and tomorrow, and tomorrow...

This allows printing text separately from the music, which is particularly useful when the input file contains several music pieces, as described in Secció 3.1.2 [Multiple scores in a book], pàgina 465.

```
\score {
  c'1
}
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
\score {
  c'1
}
```



Tomorrow, and tomorrow, and tomorrow...



Separate text blocks can be spread over multiple pages, making it possible to print text documents or books entirely within LilyPond. This feature, and the specific syntax it requires, are described in [Multi-page markup], pàgina 250.

Instruccions predefinides

\markup, \markuplist.

Fragments de codi seleccionats

Stand-alone two-column markup

Stand-alone text may be arranged in several columns using \markup commands:

```
\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
in quo Christus sumitur,
recolitur memoria passionis ejus,
mens impletur gratia,
futurae gloriae nobis pignus datur.
Amen.

*O sacred feast
in which Christ is received,
the memory of His Passion is renewed,
the mind is filled with grace,
and a pledge of future glory is given to us.
Amen.*

Vegeu també

Notation Reference: Secció 1.8.2 [Formatting text], pàgina 237, Secció 3.1.5 [File structure], pàgina 468, Secció 3.1.2 [Multiple scores in a book], pàgina 465, [Multi-page markup], pàgina 250.

Snippets: Secció “Text” in *Fragments de codi*.

Internals Reference: Secció “TextScript” in *Referència de funcionament intern*.

1.8.2 Formatting text

This section presents basic and advanced text formatting, using the \markup mode specific syntax.

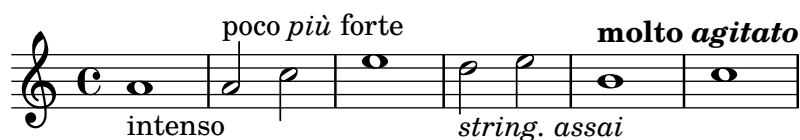
Text markup introduction

A \markup block is used to typeset text with an extensible syntax called “markup mode”.

The markup syntax is similar to LilyPond's usual syntax: a `\markup` expression is enclosed in curly braces `{...}`. A single word is regarded as a minimal expression, and therefore does not need to be enclosed with braces.

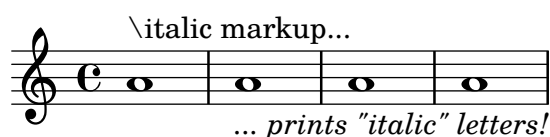
Unlike simple "quoted text" indications, `\markup` blocks may contain nested expressions or markup commands, entered using the backslash `\` character. Such commands only affect the first following expression.

```
\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
}
```



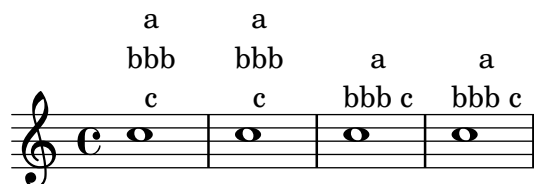
A `\markup` block may also contain quoted text strings. Such strings are treated as minimal text expressions, and therefore any markup command or special character (such as `\` and `#`) will be printed verbatim without affecting the formatting of the text. Double quotation marks themselves may be printed by preceding them with backslashes.

```
\relative {
  a'1^\markup { \italic markup... }
  a_\markup { \italic "... prints \"italic\" letters!" }
  a a
}
```



To be treated as a distinct expression, a list of words needs to be enclosed with double quotes or preceded by a command. The way markup expressions are defined affects how these expressions will be stacked, centered and aligned; in the following example, the second `\markup` expression is treated the same as the first one:

```
\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" } }
}
```



Markups can be stored in variables. Such variables may be directly attached to notes:

```
allegro = \markup { \bold \large Allegro }

{
  d''8.^~\allegro
  d'16 d'4 r2
}
```



An exhaustive list of `\markup`-specific commands can be found in Secció A.11 [Text markup commands], pàgina 674.

Vegeu també

Notation Reference: Secció A.11 [Text markup commands], pàgina 674.

Snippets: Secció “Text” in *Fragments de codi*.

Installed Files: `scm/markup.scm`.

Advertiments i problemes coneguts

Syntax error messages for markup mode can be confusing.

Selecting font and font size

Basic font switching is supported in markup mode:

```
\relative {
  d''1^~\markup {
    \bold { Più mosso }
    \italic { non troppo \underline Vivo }
  }
  r2 r4 r8
  d, _\markup { \italic quasi \smallCaps Tromba }
  f1 d2 r
}
```



The font size can be altered, relative to the global staff size, in a number of different ways.

It can be set to predefined size.

```
\relative b' {
  b1_\markup { \huge Sinfonia }
  b1^~\markup { \teeny da }
  b1-\markup { \normalsize camera }
}
```



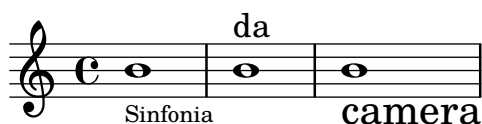
It can be set relative to its previous value.

```
\relative b' {
  b1_\markup { \larger Sinfonia }
  b1^\markup { \smaller da }
  b1-\markup { \magnify #0.6 camera }
}
```



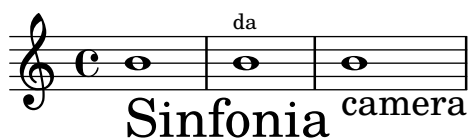
It can be increased or decreased relative to the value set by the global staff size.

```
\relative b' {
  b1_\markup { \fontsize #-2 Sinfonia }
  b1^\markup { \fontsize #1 da }
  b1-\markup { \fontsize #3 camera }
}
```



It can also be set to a fixed point-size, regardless of the global staff size.

```
\relative b' {
  b1_\markup { \abs-fontsize #20 Sinfonia }
  b1^\markup { \abs-fontsize #8 da }
  b1-\markup { \abs-fontsize #14 camera }
}
```



If the text includes spaces, then it is best to put it all inside quote marks, so that the size of each space is appropriate for the size of the other characters.

```
\markup \fontsize #6 \bold { Sinfonia da camera }
\markup \fontsize #6 \bold { "Sinfonia da camera" }
```

Sinfonia da camera

Sinfonia da camera

Text may be printed as subscript or superscript. By default these are printed in a smaller size, but a normal size can be used as well:

```
\markup {
  \column {
    \line { 1 \super st movement }
    \line { 1 \normal-size-super st movement }
    \sub { (part two) } } }
```

```
}
}
```

```
1st movement
1st movement(part two)
```

The markup mode provides an easy way to select alternate font families. The default serif font, of roman type, is automatically selected unless specified otherwise; on the last line of the following example, there is no difference between the first and the second word.

```
\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.
```

Some of these font families, used for specific items such as numbers or dynamics, do not provide all characters, as mentioned in [New dynamic marks], pàgina 128, and [Manual repeat marks], pàgina 155.

When used inside a word, some font-switching or formatting commands may produce an unwanted blank space. This can easily be solved by concatenating the text elements together:

```
\markup {
  \column {
    \line {
      \concat { 1 \super st }
      movement
    }
    \line {
      \concat { \dynamic p , }
      \italic { con dolce espressione }
    }
  }
}
```

```
1st movement
p, con dolce espressione
```

An exhaustive list of font switching commands and custom font usage commands can be found in Secció A.11.1 [Font], pàgina 674.

Defining custom font sets is also possible, as explained in Secció 1.8.3 [Fonts], pàgina 251.

Instruccions predefinides

```
\teeny, \tiny, \small, \normalsize, \large, \huge, \smaller, \larger.
```

Vegeu també

Notation Reference: Secció A.11.1 [Font], pàgina 674, [New dynamic marks], pàgina 128, [Manual repeat marks], pàgina 155, Secció 1.8.3 [Fonts], pàgina 251.

Installed Files: `scm/define-markup-commands.scm`.

Snippets: Secció “Text” in *Fragments de codi*.

Internals Reference: Secció “TextScript” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

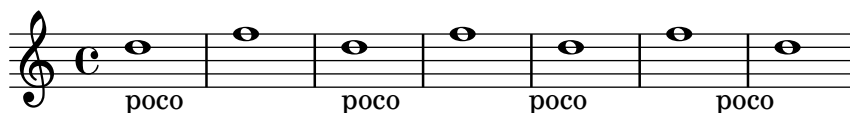
Using the font sizing commands `\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, and `\huge` will lead to inconsistent line spacing compared to using `\fontsize`.

Text alignment

This subsection discusses how to place text in markup mode. Markup objects can also be moved as a whole, using the syntax described in Secció “Moving objects” in *Manual d’aprenentatge*.

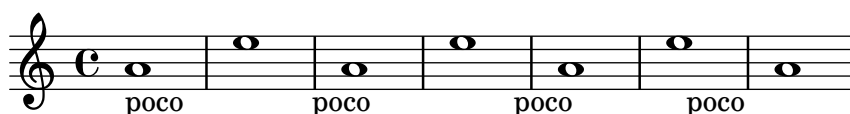
Markup objects may be aligned in different ways. By default, a text indication is aligned on its left edge: in the following example, there is no difference between the first and the second markup.

```
\relative {
  d''1-\markup { poco }
  f
  d-\markup { \left-align poco }
  f
  d-\markup { \center-align { poco } }
  f
  d-\markup { \right-align poco }
}
```



Horizontal alignment may be fine-tuned using a numeric value:

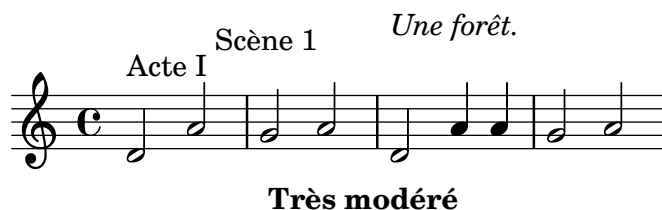
```
\relative {
  a'1-\markup { \halign #-1 poco }
  e'
  a,-\markup { \halign #0 poco }
  e'
  a,-\markup { \halign #0.5 poco }
  e'
  a,-\markup { \halign #2 poco }
}
```



Some objects may have alignment procedures of their own, and therefore are not affected by these commands. It is possible to move such markup objects as a whole, as shown for instance in [Text marks], pàgina 233.

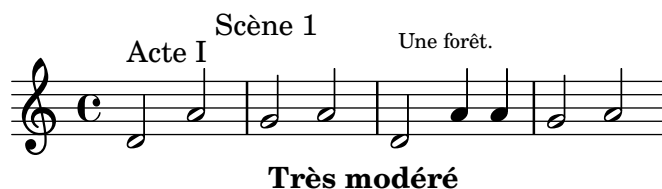
Vertical alignment is a bit more complex. As stated above, markup objects can be moved as a whole; however, it is also possible to move specific elements inside a markup block. In this case, the element to be moved needs to be preceded with an *anchor point*, that can be another markup element or an invisible object. The following example demonstrates these two possibilities; the last markup in this example has no anchor point, and therefore is not moved.

```
\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
}
```



Some commands can affect both the horizontal and vertical alignment of text objects in markup mode. Any object affected by these commands must be preceded with an anchor point:

```
\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
}
```



A markup object may include several lines of text. In the following example, each element or expression is placed on its own line, either left-aligned or centered:

```
\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

Similarly, a list of elements or expressions may be spread to fill the entire horizontal line width (if there is only one element, it will be centered on the page). These expressions can, in turn, include multi-line text or any other markup expression:

```
\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

Long text indications can also be automatically wrapped accordingly to the given line width. These will be either left-aligned or justified, as shown in the following example.

```
\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 fiesta)

An exhaustive list of text alignment commands can be found in Secció A.11.2 [Align], pàgina 684.

Vegeu també

Learning Manual: Secció “Moving objects” in *Manual d’aprenentatge*.

Notation Reference: Secció A.11.2 [Align], pàgina 684, [Text marks], pàgina 233.

Installed Files: `scm/define-markup-commands.scm`.

Snippets: Secció “Text” in *Fragments de codi*.

Internals Reference: Secció “TextScript” in *Referència de funcionament intern*.

Graphic notation inside markup

Various graphic objects may be added to a score, using markup commands.

Some markup commands allow decoration of text elements with graphics, as demonstrated in the following example.

```

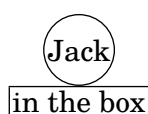
\markup \fill-line {
  \center-column {

```

```

\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

Some commands may require an increase in the padding around the text; this is achieved with some markup commands exhaustively described in Secció A.11.2 [Align], pàgina 684.

```

\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

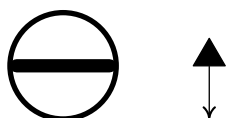
Largo to Presto

String quartet keeps very even time, Flute quartet keeps very uneven time.

Other graphic elements or symbols may be printed without requiring any text. As with any markup expression, such objects can be combined.

```
\markup {
  \combine
    \draw-circle #4 #0.4 ##f
    \filled-box #'(-4 . 4) #'(-0.5 . 0.5) #1
  \hspace #5

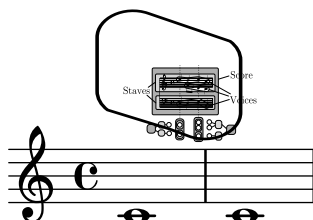
  \center-column {
    \triangle ##t
    \combine
      \draw-line #'(0 . 4)
      \arrow-head #Y #DOWN ##f
  }
}
```



Advanced graphic features include the ability to include external image files converted to the Encapsulated PostScript format (*eps*), or to directly embed graphics into the input file, using native PostScript code. In such a case, it may be useful to explicitly specify the size of the drawing, as demonstrated below:

```
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'



An exhaustive list of graphics-specific commands can be found in Secció A.11.3 [Graphic], pàgina 699.

Vegeu també

Notation Reference: Secció A.11.2 [Align], pàgina 684, Secció 5.4.4 [Dimensions], pàgina 604, Secció 1.7 [Editorial annotations], pàgina 215, Secció A.11.3 [Graphic], pàgina 699.

Installed Files: `scm/define-markup-commands.scm`, `scm/stencil.scm`.

Snippets: Secció “Text” in *Fragments de codi*.

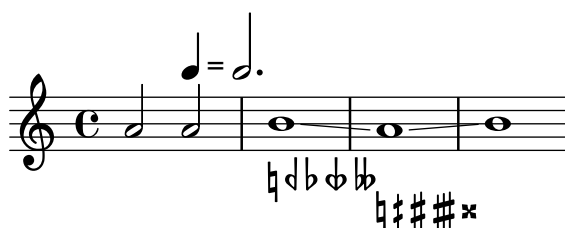
Internals Reference: Secció “TextScript” in *Referència de funcionament intern*.

Music notation inside markup

Various musical notation elements may be added to a score, inside a markup object.

Notes and accidentals can be entered using markup commands:

```
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'
```



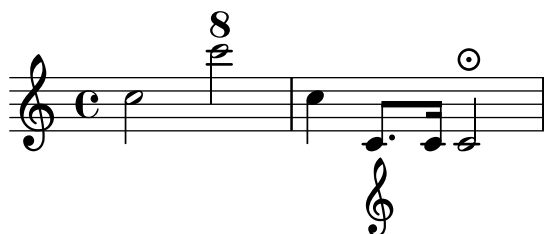
Other notation objects may also be printed in markup mode:

```
\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
  }
}
```



More generally, any available musical symbol may be included separately in a markup object, as demonstrated below; an exhaustive list of these symbols and their names can be found in Secció A.8 [The Feta font], pàgina 651.

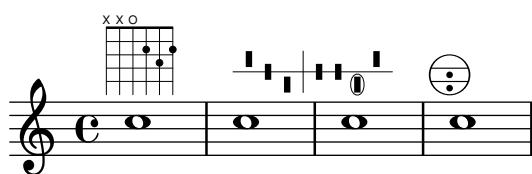
```
\relative {
  c''2
  c'^\markup { \musicglyph #"eight" }
  c,4
  c,8._\markup { \musicglyph #"clefs.G_change" }
  c16
  c2^\markup { \musicglyph #"timesig.neomensural94" }
}
```



Another way of printing non-text glyphs is described in [Fonts explained], pàgina 251. This is useful for printing braces of various sizes.

The markup mode also supports diagrams for specific instruments:

```
\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"
  }
}
```



Such diagrams are documented in Secció A.11.5 [Instrument Specific Markup], pàgina 713.

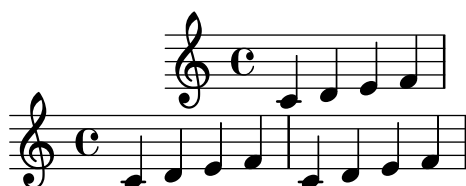
A whole score can even be nested inside a markup object. In such a case, the nested `\score` block must contain a `\layout` block, as demonstrated here:

```
\relative {
```

```

c'4 d^\markup {
  \score {
    \relative { c'4 d e f }
    \layout { }
  }
}
e f |
c d e f
}

```



An exhaustive list of music notation related commands can be found in Secció A.11.4 [Music], pàgina 707.

Vegeu també

Notation Reference: Secció A.11.4 [Music], pàgina 707, Secció A.8 [The Feta font], pàgina 651, [Fonts explained], pàgina 251.

Installed Files: `scm/define-markup-commands.scm`, `scm/fret-diagrams.scm`, `scm/harp-pedals.scm`.

Snippets: Secció “Text” in *Fragments de codi*.

Internals Reference: Secció “TextScript” in *Referència de funcionament intern*.

Multi-page markup

Although standard markup objects are not breakable, a specific syntax makes it possible to enter lines of text that can spread over multiple pages:

```

\markuplist {
  \justified-lines {
    A very long text of justified lines.
    ...
  }
  \wordwrap-lines {
    Another very long paragraph.
    ...
  }
  ...
}

```

A very long text of justified lines. ...

Another very long paragraph. ...

...

This syntax accepts a list of markups, that can be

- the result of a markup list command,

- a list of markups,
- a list of markup lists.

An exhaustive list of markup list commands can be found in Secció A.12 [Text markup list commands], pàgina 728.

Vegeu també

Notation Reference: Secció A.12 [Text markup list commands], pàgina 728.

Extending LilyPond: Secció “New markup list command definition” in *Extendre*.

Installed Files: `scm/define-markup-commands.scm`.

Snippets: Secció “Text” in *Fragments de codi*.

Internals Reference: Secció “TextScript” in *Referència de funcionament intern*.

Instruccions predefinides

`\markuplist`.

1.8.3 Fonts

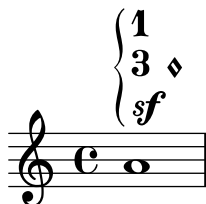
This section presents the way fonts are handled, and how they may be changed in scores.

Fonts explained

Fonts are handled through several libraries. FontConfig is used to detect available fonts on the system; the selected fonts are rendered using Pango.

Music notation fonts can be described as a set of specific glyphs, ordered in several families. The following syntax allows various LilyPond `feta` non-text fonts to be used directly in markup mode:

```
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"
  }
}
```



However, all these glyphs except the braces of various sizes contained in `fetaBraces` are available using the simpler syntax described in [Music notation inside markup], pàgina 248.

When using the glyphs contained in `fetaBraces`, the size of the brace is specified by the numerical part of the glyph name, in arbitrary units. Any integer from 0 to 575 inclusive may be specified, 0 giving the smallest brace. The optimum value must be determined by trial and error. These glyphs are all left braces; right braces may be obtained by rotation, see Secció 5.4.9 [Rotating objects], pàgina 617.

Three families of text fonts are made available:

- The *roman* (serif) font, which defaults to LilyPond Serif (an alias of TeX Gyre Schola).
- The *sans* font, which defaults to LilyPond Sans Serif (an alias of TeX Gyre Heros).
- The *typewriter* (monospaced) font, which defaults to LilyPond Monospace (an alias of TeX Gyre Cursor).

Each family may include different shapes and series. The following example demonstrates the ability to select alternate families, shapes, series and sizes. The value supplied to **font-size** is the required change from the default size.

```
\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
```



A similar syntax may be used in markup mode; however in this case it is preferable to use the simpler syntax explained in [Selecting font and font size], pàgina 239:

```
\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

Although it is easy to switch between preconfigured fonts, it is also possible to use other fonts, as explained in the following sections: [Single entry fonts], pàgina 253, and [Entire document fonts], pàgina 253.

Vegeu també

Notation Reference: Secció A.8 [The Feta font], pàgina 651, [Music notation inside markup], pàgina 248, Secció 5.4.9 [Rotating objects], pàgina 617, [Selecting font and font size], pàgina 239, Secció A.11.1 [Font], pàgina 674.

Single entry fonts

Any font that is installed on the operating system and recognized by FontConfig may be used in a score, using the following syntax:

```
\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 can be described using a comma-separated list of ‘fonts’ and a white-space separated list of ‘styles’. As long as the ‘font’ in the list is installed and contains requested glyph, it will be used, otherwise the *next* font in the list will be used instead.

Running lilypond with the following option displays a list of all available fonts on the operating system:

```
lilypond -dshow-available-fonts x
```

Vegeu també

Notation Reference: [Fonts explained], pàgina 251, [Entire document fonts], pàgina 253.

Snippets: Secció “Text” in *Fragments de codi*.

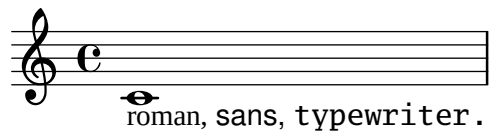
Entire document fonts

It is possible to change the fonts to be used as the default fonts in the *roman*, *sans* and *type-writer* font families by specifying them, in that order, as shown in the example below, which automatically scales the fonts with the value set for the global staff size. Similar to [Single entry fonts], pàgina 253, it can be described using a comma-separated list of ‘fonts’. However, font ‘styles’ can not be described. For an explanation of fonts, see [Fonts explained], pàgina 251.

```
\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. }  
}
```



Vegeu també

Notation Reference: [Fonts explained], pàgina 251, [Single entry fonts], pàgina 253, [Selecting font and font size], pàgina 239, Secció A.11.1 [Font], pàgina 674.

Vegeu també

Music Glossary: Secció “ambitus” in *Glossari musical*.

Learning Manual: Secció “Setting simple songs” in *Manual d’aprenentatge*.

Notation Reference: [Text markup introduction], pàgina 237, [\[Ambitus\]](#), pàgina [\[undefined\]](#), [Score layouts for choral], pàgina 295.

Snippets: Secció “Vocal music” in *Fragments de codi*.

Entering lyrics

Lyrics are entered in a special input mode, which can be introduced by the keyword `\lyricmode`, or by using `\addlyrics` or `\lyricsto`. In this special input mode, the input `d` is not parsed as the pitch *D*, but rather as a one-letter syllable of text. In other words, syllables are entered like notes but with pitches replaced by text.

For example:

```
\lyricmode { Three4 blind mice,2 three4 blind mice2 }
```

There are two main methods for specifying the horizontal placement of the syllables, either by specifying the duration of each syllable explicitly, as in the example above, or by leaving the lyrics to be aligned automatically to a melody or other voice of music, using `\addlyrics` or `\lyricsto`. The former method is described below in [Manual syllable durations], pàgina 261. The latter method is described in [Automatic syllable durations], pàgina 259.

A word or syllable of lyrics begins with an alphabetic character (plus some other characters, see below) and is terminated by any white space or a digit. Later characters in the syllable can be any character that is not a digit or white space.

Because any character that is not a digit or white space is regarded as part of the syllable, a word is valid even if it ends with `}`, which often leads to the following mistake:

```
\lyricmode { lah lah lah}
```

In this example, the `}` is included in the final syllable, so the opening brace is not balanced and the input file will probably not compile. Instead, braces should always be surrounded with white space:

```
\lyricmode { lah lah lah }
```

Punctuation, lyrics with accented characters, characters from non-English languages, or special characters (such as the heart symbol or slanted quotes), may simply be inserted directly into the input file, providing it is saved with UTF-8 encoding. For more information, see Secció 3.3.3 [Special characters], pàgina 502.

```
\relative { d''8 c16 a bes8 f ees' d c4 }
\addlyrics { „Schad' um das schö -- ne grü -- ne Band, }
```



Normal quotes may be used in lyrics, but they have to be preceded with a backslash character and the whole syllable has to be enclosed between additional quotes. For example,

```
\relative { \time 3/4 e'4 e4. e8 d4 e d c2. }
\addlyrics { "\"I" am so lone -- "ly,\"" said she }
```



The full definition of a word start in lyrics mode is somewhat more complex. A word in lyrics mode is one that begins with an alphabetic character, `_`, `?`, `!`, `:`, `'`, the control characters `^A` through `^F`, `^Q` through `^W`, `^Y`, `^_`, any 8-bit character with an ASCII code over 127, or a two-character combination of a backslash followed by one of ```, `'`, `"`, or `^`.

Great control over the appearance of lyrics comes from using `\markup` inside the lyrics themselves. For explanation of many options, see Secció 1.8.2 [Formatting text], pàgina 237.

Fragments de codi seleccionats

Formatting lyrics syllables

Markup mode may be used to format individual syllables in lyrics.

```
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
>>
```



Vegeu també

Learning Manual: Secció “Songs” in *Manual d’aprenentatge*.

Notation Reference: [Automatic syllable durations], pàgina 259, Secció 1.8.3 [Fonts], pàgina 251, Secció 1.8.2 [Formatting text], pàgina 237, Secció 5.4.1 [Input modes], pàgina 601, [Manual syllable durations], pàgina 261, Secció 3.3.3 [Special characters], pàgina 502.

Internals Reference: Secció “LyricText” in *Referència de funcionament intern*.

Snippets: Secció “Text” in *Fragments de codi*.

Aligning lyrics to a melody

Lyrics are interpreted in `\lyricmode` and printed in a `Lyrics` context, see Secció 5.1.1 [Contexts explained], pàgina 571.

```
\new Lyrics \lyricmode { ... }
```

Two variants of `\lyricmode` additionally set an associated context used to synchronise the lyric syllables to music. The more convenient `\addlyrics` immediately follows the musical content of the Voice context with which it should be synchronised, implicitly creating a Lyrics context of its own. The more versatile `\lyricsto` requires both specifying the associated Voice context by name and explicitly creating a containing Lyrics context. For details see [Automatic syllable durations], pàgina 259.

Lyrics can be aligned with melodies in two main ways:

- Lyrics can be aligned automatically, with the durations of the syllables being taken from another voice of music or (in special circumstances) an associated melody, using `\addlyrics`,

`\lyricsto`, or by setting the `associatedVoice` property. For more details, see [Automatic syllable durations], pàgina 259.

```
<<
  \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
    }
  >>

% takes durations and alignment from notes in "one"
\new Lyrics \lyricsto "one" {
  Life is __ _ love, live __ life.
}

% takes durations and alignment from notes in "one" initially
% then switches to "two"
\new Lyrics \lyricsto "one" {
  No more let
  \set associatedVoice = "two" % must be set one syllable early
  sins and sor -- rows grow.
}
>>
```



The first stanza shows the normal way of entering lyrics.

The second stanza shows how the voice from which the lyric durations are taken can be changed. This is useful if the words to different stanzas fit the notes in different ways and all the durations are available in Voice contexts. For more details, see Secció 2.1.3 [Stanzas], pàgina 286.

- Lyrics can be aligned independently of the duration of any notes if the durations of the syllables are specified explicitly, and entered with `\lyricmode`.

```
<<
  \new Voice = "one" \relative {
    \time 2/4
    c''4 b8. a16 g4. f8 e4 d c2
  }

% uses previous explicit duration of 2;
\new Lyrics \lyricmode {
```

```

    Joy to the earth!
}

% explicit durations, set to a different rhythm
\new Lyrics \lyricmode {
    Life4 is love,2. live4 life.2
}
>>

```



The first stanza is not aligned with the notes because the durations were not specified, and the previous value of 2 is used for each word.

The second stanza shows how the words can be aligned quite independently from the notes. This is useful if the words to different stanzas fit the notes in different ways and the required durations are not available in a music context. For more details see [Manual syllable durations], pàgina 261. This technique is also useful when setting dialogue over music; for examples showing this, see [Dialogue over music], pàgina 303.

Vegeu també

Learning Manual: Secció “Aligning lyrics to a melody” in *Manual d’aprenentatge*.

Notation Reference: Secció 5.1.1 [Contexts explained], pàgina 571, [Automatic syllable durations], pàgina 259, Secció 2.1.3 [Stanzas], pàgina 286, [Manual syllable durations], pàgina 261, [Dialogue over music], pàgina 303, [Manual syllable durations], pàgina 261.

Internals Reference: Secció “Lyrics” in *Referència de funcionament intern*.

Automatic syllable durations

Lyrics can be automatically aligned to the notes of a melody in three ways:

- by specifying the named Voice context containing the melody with `\lyricsto`,
- by introducing the lyrics with `\addlyrics` and placing them immediately after the Voice context containing the melody,
- by setting the `associatedVoice` property, the alignment of the lyrics may be switched to a different named Voice context at any musical moment.

In all three methods hyphens can be drawn between the syllables of a word and extender lines can be drawn beyond the end of a word. For details, see [Extenders and hyphens], pàgina 267.

The Voice context containing the melody to which the lyrics are being aligned must not have “died”, or the lyrics after that point will be lost. This can happen if there are periods when that voice has nothing to do. For methods of keeping contexts alive, see Secció 5.1.3 [Keeping contexts alive], pàgina 576.

Using `\lyricsto`

Lyrics can be aligned under a melody automatically by specifying the named Voice context containing the melody with `\lyricsto`:

```

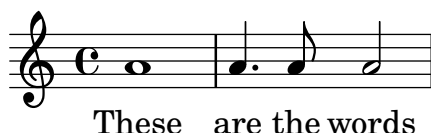
<<
\new Voice = "melody" \relative {

```

```

    a'1 a4. a8 a2
  }
  \new Lyrics \lyricsto "melody" {
    These are the words
  }
>>

```



This aligns the lyrics to the notes of the named **Voice** context, which must already exist. Therefore normally the **Voice** context is specified first, followed by the **Lyrics** context. The lyrics themselves follow the `\lyricsto` command. The `\lyricsto` command invokes lyric mode automatically. By default, the lyrics are placed underneath the notes. For other placements, see [Placing lyrics vertically], pàgina 269.

Using `\addlyrics`

The `\addlyrics` command is just a convenient shortcut that can sometimes be used instead of having to set up the lyrics through a more complicated LilyPond structure.

```

{ MUSIC }
\addlyrics { LYRICS }

```

is the same as

```

\new Voice = "blah" { MUSIC }
\new Lyrics \lyricsto "blah" { LYRICS }

```

Here is an example,

```

{
  \time 3/4
  \relative { c'2 e4 g2. }
  \addlyrics { play the game }
}

```



More stanzas can be added by adding more `\addlyrics` sections:

```

{
  \time 3/4
  \relative { c'2 e4 g2. }
  \addlyrics { play the game }
  \addlyrics { speel het spel }
  \addlyrics { joue le jeu }
}

```



The command `\addlyrics` cannot handle polyphonic settings. Also, it cannot be used to associate lyrics to a `TabVoice`. For these cases one should use `\lyricsto`.

Using `associatedVoice`

The melody to which the lyrics are being aligned can be changed by setting the `associatedVoice` property,

```
\set associatedVoice = #"lala"
```

The value of the property (here: `"lala"`) should be the name of a `Voice` context. For technical reasons, the `\set` command must be placed one syllable before the one to which the change in voice is to apply.

Here is an example demonstrating its use:

```
<<
\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
  }
  >>
% takes durations and alignment from notes in "one" initially
% then switches to "two"
\new Lyrics \lyricsto "one" {
  No more let
  \set associatedVoice = "two" % must be set one syllable early
  sins and sor -- rows grow.
}
>>
```



Vegeu també

Notation Reference: [Extenders and hyphens], pàgina 267, Secció 5.1.3 [Keeping contexts alive], pàgina 576, [Placing lyrics vertically], pàgina 269.

Manual syllable durations

In some complex vocal music, it may be desirable to place lyrics completely independently of notes. In this case do not use `\lyricsto` or `\addlyrics` and do not set `associatedVoice`.

Syllables are entered like notes – but with pitches replaced by text – and the duration of each syllable is entered explicitly after the syllable.

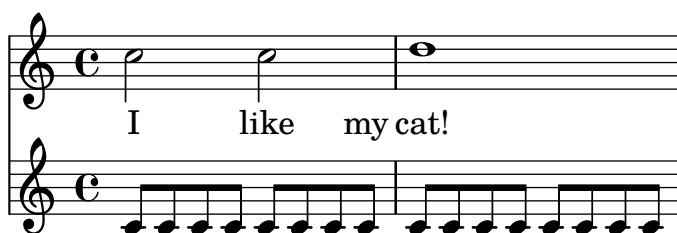
Hyphenated lines may be drawn between syllables as usual, but extender lines cannot be drawn when there is no associated voice.

Here are two examples:

```
<<
\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
  }
}
>>
```



This technique is useful when writing dialogue over music, see [Dialogue over music], pàgina 303.

To change syllable alignment, simply override the `self-alignment-X` property:

```
<<
\new Voice = "melody" \relative {
```



```

\time 3/4
c'2 e4 g2 f
}
\new Lyrics \lyricmode {
  \override LyricText.self-alignment-X = #LEFT
  play1 a4 game4
}
>>

```



Vegeu també

Notation Reference: [Dialogue over music], pàgina 303.

Internals Reference: Secció “Lyrics” in *Referència de funcionament intern*, Secció “Voice” in *Referència de funcionament intern*.

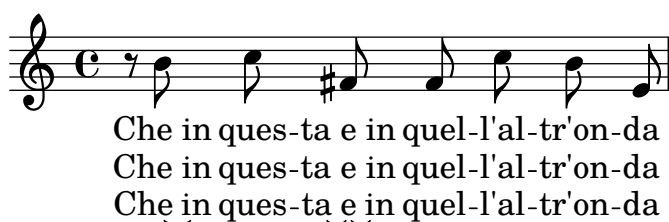
Multiple syllables to one note

In order to assign more than one syllable to a single note with spaces between the syllables, you can surround the phrase with quotes or use a _ character. Alternatively, you can use the tilde symbol (~) to get a lyric tie.

```

{
  \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 }
}

```



Vegeu també

Internals Reference: Secció “LyricCombineMusic” in *Referència de funcionament intern*.

Multiple notes to one syllable

Sometimes, particularly in Medieval and baroque music, several notes are sung on one syllable; this is called melisma, see Secció “melisma” in *Glossari musical*. The syllable to a melisma is usually left-aligned with the first note of the melisma.

When a melisma occurs on a syllable other than the last one in a word, that syllable is usually joined to the following one with a hyphenated line. This is indicated by placing a double hyphen, --, immediately after the syllable.

Alternatively, when a melisma occurs on the last or only syllable in a word an extender line is usually drawn from the end of the syllable to the last note of the melisma. This is indicated by placing a double underscore, __, immediately after the word.

There are five ways in which melismata can be indicated:

- Melismata are created automatically over notes which are tied together:

```
<<
\new Voice = "melody" \relative {
  \time 3/4
  f''4 g2 ~ |
  4 e2 ~ |
  8
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- e __
}
>>
```



- Melismata can be created automatically from the music by placing slurs over the notes of each melisma. This is the usual way of entering lyrics:

```
<<
\new Voice = "melody" \relative {
  \time 3/4
  f''4 g8 ( f e f )
  e8 ( d e2 )
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- e __
}
>>
```



Note that phrasing slurs do not affect the creation of melismata.

- Notes are considered a melisma if they are manually beamed, providing automatic beaming is switched off. See [Setting automatic beam behavior], pàgina 84.

```
<<
```

```

\new Voice = "melody" \relative {
  \time 3/4
  \autoBeamOff
  f''4 g8[ f e f]
  e2.
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- e
}
>>

```



Clearly this is not suited to melismata over notes which are longer than eighth notes.

- An unslurred group of notes will be treated as a melisma if they are bracketed between `\melisma` and `\melismaEnd`.

```

<<
\new Voice = "melody" \relative {
  \time 3/4
  f''4 g8
  \melisma
  f e f
  \melismaEnd
  e2.
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- e
}
>>

```



- A melisma can be defined entirely in the lyrics by entering a single underscore character, `_`, for every extra note that has to be added to the melisma.

```

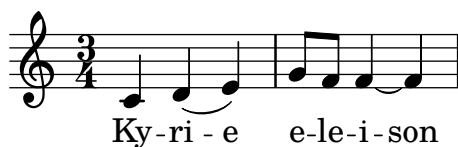
<<
\new Voice = "melody" \relative {
  \time 3/4
  f''4 g8 f e f
  e8 d e2
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- _ _ _ e _ _ _
}
>>

```



It is possible to have ties, slurs and manual beams in the melody without their indicating melismata. To do this, set `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
}
>>
```

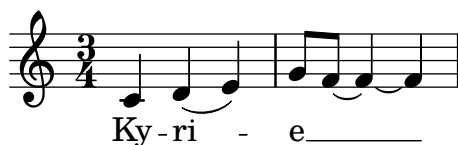


Other settings for `melismaBusyProperties` can be used to selectively include or exclude ties, slurs, and beams from the automatic detection of melismata; see `melismaBusyProperties` in Secció “Tunable context properties” in *Referència de funcionament intern*.

Alternatively, if all melismata indications are to be ignored, `ignoreMelismata` may be set true; see [Stanzas with different rhythms], pàgina 287.

If a melisma is required during a passage in which `melismaBusyProperties` is active, it may be indicated by placing a single underscore in the lyrics for each note which should be included in the melisma:

```
<<
\new Voice = "melody" \relative {
  \time 3/4
  \set melismaBusyProperties = #'()
  c'4 d ( e )
  g8 [ f ] ~ 4 ~ f
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- _ e _ _ _ _
}
>>
```



Instruccions predefinides

`\autoBeamOff`, `\autoBeamOn`, `\melisma`, `\melismaEnd`.

Vegeu també

Musical Glossary: Secció “melisma” in *Glossari musical*.

Learning Manual: Secció “Aligning lyrics to a melody” in *Manual d’aprenentatge*.

Notation Reference: [Aligning lyrics to a melody], pàgina 257, [Automatic syllable durations], pàgina 259, [Setting automatic beam behavior], pàgina 84, [Stanzas with different rhythms], pàgina 287.

Internals Reference: Secció “Tunable context properties” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Extender lines under melismata are not created automatically; they must be inserted manually with a double underscore.

Extenders and hyphens

In the last syllable of a word, melismata are sometimes indicated with a long horizontal line starting in the melisma syllable, and ending in the next one. Such a line is called an extender line, and it is entered as ‘`--`’ (note the spaces before and after the two underscore characters).

Nota: Melismata are indicated in the score with extender lines, which are entered as one double underscore; 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.

Centered hyphens are entered as ‘`--`’ between syllables of a same word (note the spaces before and after the two hyphen characters). The hyphen will be centered between the syllables, and its length will be adjusted depending on the space between the syllables.

In tightly engraved music, hyphens can be removed. Whether this happens can be controlled with the `minimum-distance` (minimum distance between two syllables) and the `minimum-length` (threshold below which hyphens are removed) properties of `LyricHyphen`.

Vegeu també

Internals Reference: Secció “LyricExtender” in *Referència de funcionament intern*, Secció “LyricHyphen” in *Referència de funcionament intern*.

2.1.2 Techniques specific to lyrics

Working with lyrics and variables

Variables containing lyrics can be created, but the lyrics must be entered in lyric mode:

```
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
}
>>
}

```



Durations do not need to be added if the variable is to be invoked with `\addlyrics` or `\lyricsto`.

For different or more complex orderings, the best way is to define the music and lyric variables first, then set up the hierarchy of staves and lyrics, omitting the lyrics themselves, and then add the lyrics using `\context` underneath. This ensures that the voices referenced by `\lyricsto` have always been defined earlier. For example:

```

sopranoMusic = \relative { c'4 c c c }
contraltoMusic = \relative { a4 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
      }
    }
  }
  >>
}

```



Vegeu també

Notation Reference: [Placing lyrics vertically], pàgina 269.

Internals Reference: Secció “LyricCombineMusic” in *Referència de funcionament intern*, Secció “Lyrics” in *Referència de funcionament intern*.

Placing lyrics vertically

Depending on the type of music, lyrics may be positioned above the staff, below the staff, or between staves. Placing lyrics below the associated staff is the easiest, and can be achieved by simply defining the Lyrics context below the Staff context:

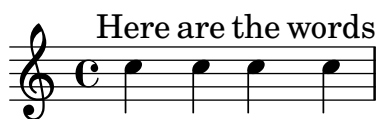
```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



Lyrics may be positioned above the staff using one of two methods. The simplest (and preferred) method is to use the same syntax as above and explicitly specify the position of the lyrics:

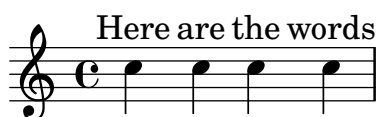
```
\score {
  <<
    \new Staff = "staff" {
      \new Voice = "melody" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics \with { alignAboveContext = "staff" } {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```

}



Alternatively, a two-step process may be used. First the Lyrics context is declared (without any content) before the Staff and Voice contexts, then the `\lyricsto` command is placed after the Voice declaration it references by using `\context`, as follows:

```
\score {
  <<
    \new Lyrics = "lyrics" \with {
      % lyrics above a staff should have this 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
      }
    }
  >>
}
```



When there are two voices on separate staves the lyrics may be placed between the staves using either of these methods. Here is an example of the second method:

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice = "sopranos" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics = "sopranos"
    \new Lyrics = "contraltos" \with {
      % lyrics above a staff should have this 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
  }
}
>>
}

```



Other combinations of lyrics and staves may be generated by elaborating these examples, or by examining the templates in the Learning Manual, see Secció “Vocal ensembles templates” in *Manual d’aprenentatge*.

Fragments de codi seleccionats

Obtaining 2.12 lyrics spacing in newer versions

The vertical spacing engine changed since version 2.14. This can cause lyrics to be spaced differently.

It is possible to set properties for **Lyric** and **Staff** contexts to get the spacing engine to behave as it did in version 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))
}
}
}

```

The image displays a musical score for the hymn "Great is Thy faithful-ness". It consists of two systems of music, each with a treble and bass staff. The first system shows the lyrics "Great is Thy" aligned with the notes. The second system shows the lyrics "faith - - - ful - ness," with hyphens indicating syllable continuation. The score is rendered with a specific staff spacing and lyric alignment, demonstrating the effect of the provided LilyPond code.

Vegeu també

Learning Manual: Secció “Vocal ensembles templates” in *Manual d’aprenentatge*.

Notation Reference: Secció 5.1.7 [Context layout order], pàgina 588, Secció 5.1.2 [Creating and referencing contexts], pàgina 573.

Placing syllables horizontally

To increase the spacing between lyrics, set the `minimum-distance` property of `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
}
```



To make this change for all lyrics in the score, set the property in the `\layout` block.

```
\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
    }
  }
}
```



Fragments de codi seleccionats

Lyrics alignment

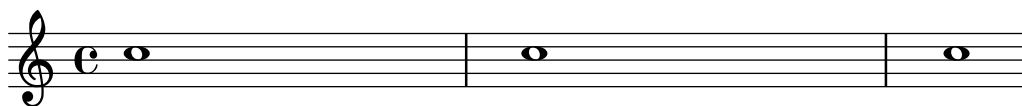
Horizontal alignment for lyrics can be set by overriding the `self-alignment-X` property of the `LyricText` object. `#-1` is left, `#0` is center and `#1` is right; however, you can use `#LEFT`, `#CENTER` and `#RIGHT` as well.

```
\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 = #RIGHT
  "This is right-aligned"
}

```



This is left-aligned This is centered This is right-aligned

Checking to make sure that text scripts and lyrics are within the margins requires additional calculations. To speed up processing slightly, this feature can be disabled:

```
\override Score.PaperColumn.keep-inside-line = ##f
```

To make lyrics avoid bar lines as well, use

```

\layout {
  \context {
    \Lyrics
    \consists "Bar_engraver"
    \consists "Separating_line_group_engraver"
    \hide BarLine
  }
}

```

Lyrics and repeats

Simple repeats

Repeats in *music* are fully described elsewhere; see Secció 1.4 [Repeats], pàgina 147. This section explains how to add lyrics to repeated sections of music.

Lyrics to a section of music that is repeated should be surrounded by exactly the same repeat construct as the music, if the words are unchanged.

```

\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat volta 2 { b4 b b b }
        }
      }
    }
  \new Lyrics {
    \lyricsto "melody" {

```

```

    Not re -- peat -- ed.
    \repeat volta 2 { Re -- peat -- ed twice. }
  }
}
>>
}

```



The words will then be correctly expanded if the repeats are unfolded.

```

\score {
  \unfoldRepeats {
    <<
      \new Staff {
        \new Voice = "melody" {
          \relative {
            a'4 a a a
            \repeat volta 2 { b4 b b b }
          }
        }
      }
      \new Lyrics {
        \lyricsto "melody" {
          Not re -- peat -- ed.
          \repeat volta 2 { Re -- peat -- ed twice. }
        }
      }
    >>
  }
}

```



If the repeated section is to be unfolded and has different words, simply enter all the words:

```

\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat unfold 2 { b4 b b b }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {

```

```

    Not re -- peat -- ed.
    The first time words.
    Sec -- ond time words.
  }
}
>>
}

```



When the words to a repeated volta section are different, the words to each repeat must be entered in separate **Lyrics** contexts, correctly nested in parallel sections:

```

\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat volta 2 { b4 b b b }
        }
      }
    }
  }
  \new Lyrics \lyricsto "melody" {
    Not re -- peat -- ed.
    <<
      { The first time words. }
      \new Lyrics {
        \set associatedVoice = "melody"
        Sec -- ond time words.
      }
    }
  }
  >>
}

```



More verses may be added in a similar way:

```

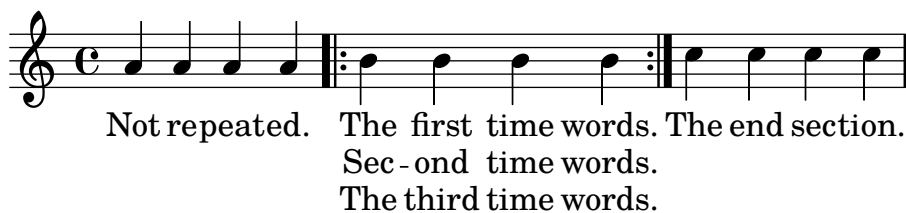
\score {
  <<
    \new Staff {
      \new Voice = "singleVoice" {
        \relative {
          a'4 a a a
          \repeat volta 3 { b4 b b b }
        }
      }
    }
  }
}

```

```

        c4 c c c
      }
    }
  }
  \new Lyrics \lyricsto "singleVoice" {
    Not re -- peat -- ed.
    <<
    { The first time words. }
    \new Lyrics {
      \set associatedVoice = "singleVoice"
      Sec -- ond time words.
    }
    \new Lyrics {
      \set associatedVoice = "singleVoice"
      The third time words.
    }
    >>
    The end sec -- tion.
  }
>>
}

```



However, if this construct is embedded within a multi-staved context such as a `ChoirStaff` the lyrics of the second and third verses will appear beneath the bottom staff.

To position them correctly use `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" {
    Not re -- peat -- ed.
    <<
    { The first time words. }
    \new Lyrics = "secondVerse"
    \with { alignBelowContext = #"firstVerse" } {
      \set associatedVoice = "melody"
      Sec -- ond time words.
    }
  }
}

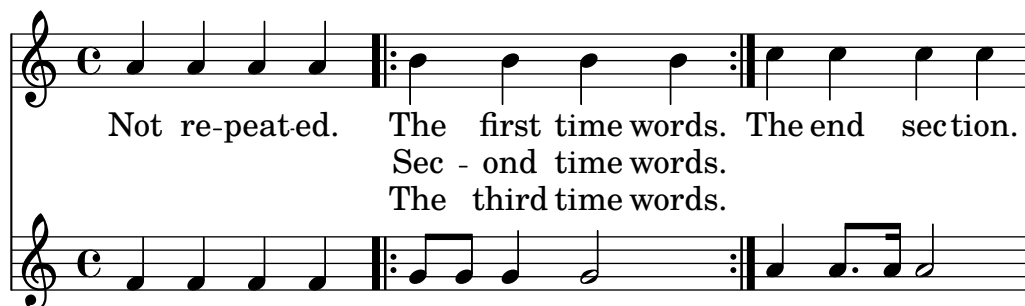
```



```

\new Lyrics = "thirdVerse"
\with { alignBelowContext = #"secondVerse" } {
  \set associatedVoice = "melody"
  The third time words.
}
>>
The end sec -- tion.
}
\new Voice = "harmony" {
  \relative {
    f'4 f f f \repeat volta 2 { g8 g g4 g2 } a4 a8. a16 a2
  }
}
>>
}

```



Not re-peated. The first time words. The end section.
 Sec - ond time words.
 The third time words.

Repeats with alternative endings

If the words of the repeated section are the same, and none of the `\alternative` blocks start with a rest, exactly the same structure can be used for both the lyrics and music. This has the advantage that `\unfoldRepeats` will expand both music and lyrics correctly.

```

\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" {
      Not re -- peat -- ed.
      \repeat volta 2 { Re -- peat -- }
      \alternative { { ed twice. } { ed twice. } }
    }
  }
  >>
}

```



But when the repeated section has different words, or when one of the `\alternative` blocks starts with a rest, a repeat construct cannot be used around the words and `\skip` commands have to be inserted manually to skip over the notes in the alternative sections which do not apply.

Note: do not use an underscore, `_`, to skip notes – an underscore indicates a melisma, causing the preceding syllable to be left-aligned.

Nota: The `\skip` command must be followed by a number, but this number is ignored in lyrics which derive their durations from the notes in an associated melody through `\addlyrics` or `\lyricsto`. Each `\skip` skips a single note of any value, irrespective of the value of the following number.

```
\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" {
        The first time words.
        \repeat unfold 2 { \skip 1 }
        End here.
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Sec -- ond
        \repeat unfold 2 { \skip 1 }
        time words.
      }
    }
  >>
}
```



When a note is tied over into two or more alternative endings a tie is used to carry the note into the first alternative ending and a `\repeatTie` is used in the second and subsequent endings.

This structure causes difficult alignment problems when lyrics are involved and increasing the length of the alternative sections so the tied notes are contained wholly within them may give a more acceptable result.

The tie creates a melisma into the first alternative, but not into the second and subsequent alternatives, so to align the lyrics correctly it is necessary to disable the automatic creation of melismata over the volta section and insert manual skips.

```
\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 { Here's a __ }
        \alternative {
          { \skip 1 verse }
          { \skip 1 sec }
        }
        ond one.
      }
    }
  >>
}
```



Note that if `\unfoldRepeats` is used around a section containing `\repeatTie`, the `\repeatTie` should be removed to avoid both types of tie being printed.

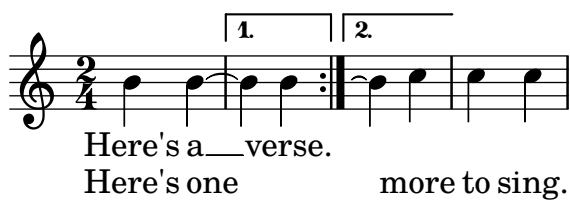
When the repeated section has different words a `\repeat` cannot be used around the lyrics and `\skip` commands need to be inserted manually, as before.

```
\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" {
    Here's a __ verse.
    \repeat unfold 2 { \skip 1 }
  }
}
\new Lyrics {
  \lyricsto "melody" {
    Here's one
    \repeat unfold 2 { \skip 1 }
    more to sing.
  }
}
>>
}

```



If you wish to show extenders and hyphens into and out of alternative sections these must be inserted manually.

```

\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" {
      Here's a __ verse.
      \repeat unfold 2 { \skip 1 }
    }
  }
  \new Lyrics {
    \lyricsto "melody" {
      Here's "a_"
      \skip 1
      "_" sec -- ond one.
    }
  }
}

```

```

    }
  >>
}

```



Vegeu també

Notation Reference: Secció 5.1.3 [Keeping contexts alive], pàgina 576, Secció 1.4 [Repeats], pàgina 147.

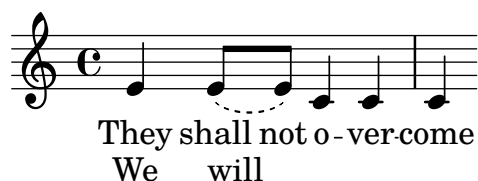
Divisi lyrics

When just the words and rhythms of the two parts differ with the pitches remaining the same, temporarily turning off the automatic detection of melismata and indicating the melisma in the lyrics may be the appropriate method to use:

```

\score {
  <<
    \new Voice = "melody" {
      \relative c' {
        \set melismaBusyProperties = #'()
        \slurDown
        \slurDashed
        e4 e8 ( e ) c4 c |
        \unset melismaBusyProperties
        c
      }
    }
    \new Lyrics \lyricsto "melody" {
      They shall not o -- ver -- come
    }
    \new Lyrics \lyricsto "melody" {
      We will _
    }
  >>
}

```



When both music and words differ it may be better to display the differing music and lyrics by naming voice contexts and attaching lyrics to those specific contexts:

```

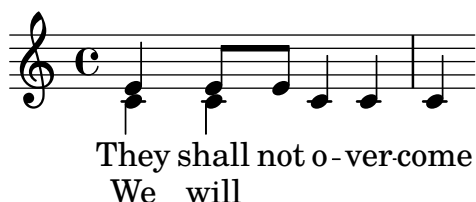
\score {
  <<
    \new Voice = "melody" {

```

```

\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
}
}
>>
}

```



It is common in choral music to have a voice part split for several measures. The `<< {...} \\ {...} >>` construct, where the two (or more) musical expressions are separated by double backslashes, might seem the proper way to set the split voices. This construct, however, will assign **all** the expressions within it to **NEW Voice contexts** which will result in *no lyrics* being set for them since the lyrics will be set to the original voice context – not, typically, what one wants. The temporary polyphonic passage is the proper construct to use, see section *Temporary polyphonic passages* in [Single-staff polyphony], pàgina 169.

Polyphony with shared lyrics

When two voices with different rhythms share the same lyrics, aligning the lyrics to one of the voices may lead to problems in the other voice. For example, the second lyric extender below is too short, since the lyrics are aligned only to the top voice:

```

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

```

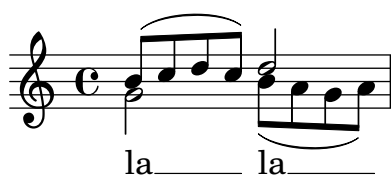
>>



To get the desired result, align the lyrics to a new `NullVoice` context containing a suitable combination of the two voices. The notes of the `NullVoice` context do not appear on the printed page, but can be used to align the lyrics appropriately:

```
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
>>
```



This method also can be used with the `\partcombine` function, which does not allow lyrics on its own:

```
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
>>
```



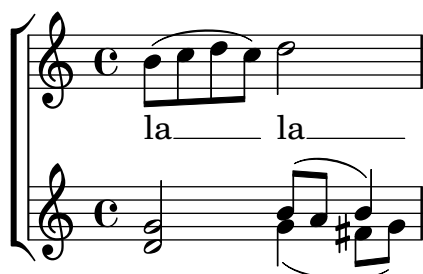
Advertiments i problemes coneguts

The `\addLyrics` function only works with `Voice` lyrics and so cannot be used with `NullVoice`. The `\partcombine` function is described in [Automatic part combining], pàgina 178.

Lastly, this method can be used even when the voices are in different staves, and is not limited to only two voices:

```
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
>>
```

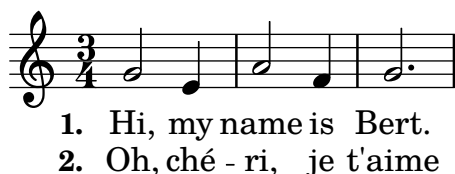


2.1.3 Stanzas

Adding stanza numbers

Stanza numbers can be added by setting `stanza`, e.g.,

```
\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
}
```



These numbers are put just before the start of the first syllable.

Adding dynamics marks to stanzas

Stanzas differing in loudness may be indicated by putting a dynamics mark before each stanza. In LilyPond, everything coming in front of a stanza goes into the `StanzaNumber` object; dynamics marks are no different. For technical reasons, you have to set the stanza outside `\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
>>

```



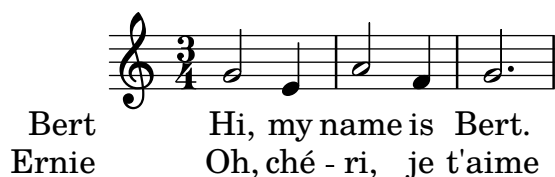
Adding singers' names to stanzas

Names of singers can also be added. They are printed at the start of the line, just like instrument names. They are created by setting `vocalName`. A short version may be entered as `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
}

```



Stanzas with different rhythms

Often, different stanzas of one song are put to one melody in slightly differing ways. Such variations can still be captured with `\lyricsto`.

Ignoring melismata

One possibility is that the text has a melisma in one stanza, but multiple syllables in another. One solution is to make the faster voice ignore the melisma. This is done by setting `ignoreMelismata` in the Lyrics context.

```

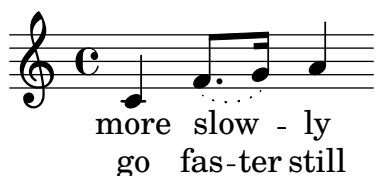
<<
\relative \new Voice = "lahlah" {
  \set Staff.autoBeaming = ##f
  c'4

```

```

\slurDotted
f8.[( g16])
a4
}
\new Lyrics \lyricsto "lahlah" {
  more slow -- ly
}
\new Lyrics \lyricsto "lahlah" {
  go
  \set ignoreMelismata = ##t
  fas -- ter
  \unset ignoreMelismata
  still
}
>>

```



Advertiments i problemes coneguts

Unlike most `\set` commands, `\set ignoreMelismata` does not work if prefixed with `\once`. It is necessary to use `\set` and `\unset` to bracket the lyrics where melismata are to be ignored.

Adding syllables to grace notes

By default, grace notes (e.g., via `\grace`) do not get assigned syllables when using `\lyricsto`, but this behavior can be changed:

```

<<
\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.
}
>>

```



Advertiments i problemes coneguts

Like `associatedVoice`, `includeGraceNotes` needs to be set at latest one syllable before the one which is to be put under a grace note. For the case of a grace note at the very beginning of a piece of music, consider using a `\with` or `\context` block:

```
<<
  \new Voice = melody \relative c' {
    \grace { c16( d e f }
    g1) f
  }
  \new Lyrics \with { includeGraceNotes = ##t }
  \lyricsto melody {
    Ah __ fa
  }
>>
```

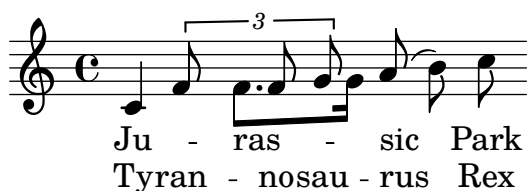


Switching to an alternative melody

More complex variations in setting lyrics to music are possible. The melody to which the lyrics are being set can be changed from within the lyrics by setting the `associatedVoice` property:

```
<<
  \relative \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
    c'4
    <<
      \new Voice = "alternative" {
        \voiceOne
        \tuplet 3/2 {
          % show associations clearly.
          \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" {
    % Tricky: need to set associatedVoice
```

```
% one syllable too soon!
\set associatedVoice = "alternative" % applies to "ran"
Ty --
ran --
no --
\set associatedVoice = "lahlah" % applies to "rus"
sau -- rus Rex
} >>
```



The text for the first stanza is set to the melody called ‘lahlah’ in the usual way, but the second stanza is set initially to the `lahlah` context and is then switched to the `alternative` melody for the syllables ‘ran’ to ‘sau’ by the lines:

```
\set associatedVoice = "alternative" % applies to "ran"
Ty --
ran --
no --
\set associatedVoice = "lahlah" % applies to "rus"
sau -- rus Rex
```

Here, `alternative` is the name of the `Voice` context containing the triplet.

Note the placement of the `\set associatedVoice` command – it appears to be one syllable too early, but this is correct.

Nota: The `\set associatedVoice` command must be placed one syllable *before* the one at which the switch to the new voice is to occur. In other words, changing the associated Voice happens one syllable later than expected. This is for technical reasons, and it is not a bug.

Printing stanzas at the end

Sometimes it is appropriate to have one stanza set to the music, and the rest added in verse form at the end of the piece. This can be accomplished by adding the extra verses into a `\markup` section outside of the main score block. Notice that there are two different ways to force linebreaks when using `\markup`.

```
melody = \relative {
e' d c d | e e e e |
d d e d | c1 |
}

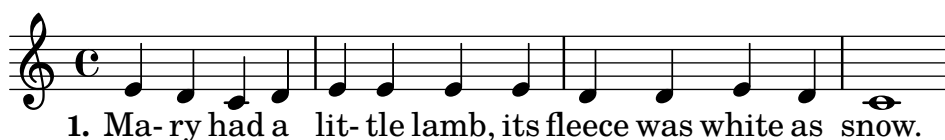
text = \lyricmode {
\set stanza = #"1." Ma- ry had a lit- tle lamb,
its fleece was white as snow.
}

\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.

Printing stanzas at the end in multiple columns

When a piece of music has many verses, they are often printed in multiple columns across the page. An outdented verse number often introduces each verse. The following example shows how to produce such 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 % moves the column off the left margin;
    % can be removed if space on the page is tight
    \column {
      \line { \bold "2."
        \column {
          "This is verse two."
          "It has two lines."
        }
      }
    }
    \combine \null \vspace #0.1 % adds vertical spacing between verses
    \line { \bold "3."
      \column {
        "This is verse three."
        "It has two lines."
      }
    }
  }
  \hspace #0.1 % adds horizontal spacing between columns;
  \column {
    \line { \bold "4."
      \column {
        "This is verse four."
        "It has two lines."
      }
    }
    \combine \null \vspace #0.1 % adds vertical spacing between verses
    \line { \bold "5."
      \column {
        "This is verse five."
        "It has two lines."
      }
    }
  }
  \hspace #0.1 % gives some extra space on the right margin;
  % can be removed if page space is tight
}
}

```



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 five.
It has two lines.

Vegeu també

Internals Reference: Secció “LyricText” in *Referència de funcionament intern*, Secció “StanzaNumber” in *Referència de funcionament intern*.

2.1.4 Songs

References for songs

Songs are usually written on three staves with the melody for the singer on the top staff and two staves of piano accompaniment at the bottom. The lyrics of the first stanza are printed immediately underneath the top staff. If there are just a small number of further stanzas these can be printed immediately under the first one, but if there are more stanzas than can be easily accommodated there the second and subsequent stanzas are printed after the music as stand-alone text.

All the notational elements needed to write songs are fully described elsewhere:

- For constructing the staff layout, see Secció 1.6.1 [Displaying staves], pàgina 186.
- For writing piano music, see Secció 2.2 [Keyboard and other multi-staff instruments], pàgina 317.
- For writing the lyrics to a melody line, see Secció 2.1.1 [Common notation for vocal music], pàgina 255.
- For placing the lyrics, see [Placing lyrics vertically], pàgina 269.
- For entering stanzas, see Secció 2.1.3 [Stanzas], pàgina 286.
- Songs are frequently printed with the chording indicated by chord names above the staves. This is described in Secció 2.7.2 [Displaying chords], pàgina 409.
- To print fret diagrams of the chords for guitar accompaniment or accompaniment by other fretted instruments, see “Fret diagram markups” in Secció 2.4.1 [Common notation for fretted strings], pàgina 332.

Vegeu també

Learning Manual: Secció “Songs” in *Manual d’aprenentatge*.

Notation Reference: Secció 2.1.1 [Common notation for vocal music], pàgina 255, Secció 2.7.2 [Displaying chords], pàgina 409, Secció 1.6.1 [Displaying staves], pàgina 186, Secció 2.2 [Keyboard and other multi-staff instruments], pàgina 317, [Placing lyrics vertically], pàgina 269, Secció 2.1.3 [Stanzas], pàgina 286.

Snippets: Secció “Vocal music” in *Fragments de codi*.

Lead sheets

Lead sheets may be printed by combining vocal parts and ‘chord mode’; this syntax is explained in Secció 2.7 [Chord notation], pàgina 404.

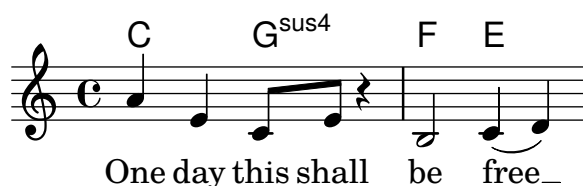
Fragments de codi seleccionats

Simple lead sheet

When put together, chord names, a melody, and lyrics form a 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 __ }
>>
```



Vegeu també

Notation Reference: Secció 2.7 [Chord notation], pàgina 404.

2.1.5 Choral

This section discusses notation issues that relate most directly to choral music. This includes anthems, part songs, oratorio, etc.

References for choral

Choral music is usually notated on two, three or four staves within a **ChoirStaff** group. Accompaniment, if required, is placed beneath in a **PianoStaff** group, which is usually reduced in size for rehearsal of *a cappella* choral works. The notes for each vocal part are placed in a **Voice** context, with each staff being given either a single vocal part (i.e., one **Voice**) or a pair of vocal parts (i.e., two **Voices**).

Words are placed in **Lyrics** contexts, either underneath each corresponding music staff, or one above and one below the music staff if this contains the music for two parts.

Several common topics in choral music are described fully elsewhere:

- An introduction to creating an SATB vocal score can be found in the Learning Manual, see Secció “Four-part SATB vocal score” in *Manual d’aprenentatge*. There is also a built-in template which simplifies the entry of SATB vocal music, see Secció “Built-in templates” in *Manual d’aprenentatge*.
- Several templates suitable for various styles of choral music can also be found in the Learning Manual, see Secció “Vocal ensembles templates” in *Manual d’aprenentatge*.
- For information about **ChoirStaff** and **PianoStaff** see [Grouping staves], pàgina 187.
- Shape note heads, as used in Sacred Harp and similar notation, are described in [Shape note heads], pàgina [undefined].
- When two vocal parts share a staff the stems, ties, slurs, etc., of the higher part will be directed up and those of the lower part down. To do this, use **\voiceOne** and **\voiceTwo**. See [Single-staff polyphony], pàgina 169.
- When a vocal part temporarily splits, you should use *Temporary polyphonic passages* (see [Single-staff polyphony], pàgina 169).

Instruccions predefinides

\oneVoice, **\voiceOne**, **\voiceTwo**.

Vegeu també

Learning Manual: Secció “Four-part SATB vocal score” in *Manual d’aprenentatge*, Secció “Vocal ensembles templates” in *Manual d’aprenentatge*.

Notation Reference: Secció 5.1.7 [Context layout order], pàgina 588, [Grouping staves], pàgina 187, [Shape note heads], pàgina [undefined], [Single-staff polyphony], pàgina 169.

Snippets: Secció “Vocal music” in *Fragments de codi*.

Internals Reference: Secció “ChoirStaff” in *Referència de funcionament intern*, Secció “Lyrics” in *Referència de funcionament intern*, Secció “PianoStaff” in *Referència de funcionament intern*.

Score layouts for choral

Choral music containing four staves, with or without piano accompaniment, is usually laid out with two systems per page. Depending on the page size, achieving this may require changes to several default settings. The following settings should be considered:

- The global staff size can be modified to change the overall size of the elements of the score. See Secció 4.2.2 [Setting the staff size], pàgina 533.
- The distances between the systems, the staves and the lyrics can all be adjusted independently. See Secció 4.4 [Vertical spacing], pàgina 541.
- The dimensions of the vertical layout variables can be displayed as an aid to adjusting the vertical spacing. This and other possibilities for fitting the music onto fewer pages are described in Secció 4.6 [Fitting music onto fewer pages], pàgina 567.
- If the number of systems per page changes from one to two it is customary to indicate this with a system separator mark between the two systems. See [Separating systems], pàgina 193.
- For details of other page formatting properties, see Secció 4.1 [Page layout], pàgina 521.

Dynamic markings by default are placed below the staff, but in choral music they are usually placed above the staff in order to avoid the lyrics. The predefined command `\dynamicUp` does this for the dynamic markings in a single `Voice` context. If there are many `Voice` contexts this predefined command would have to be placed in every one. Alternatively its expanded form can be used to place all dynamic markings in the entire score above their respective staves, as shown here:

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice {
        \relative { g'4\f 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
    }
  }
}
```



Instruccions predefinides

`\dynamicUp`, `\dynamicDown`, `\dynamicNeutral`.

Vegeu també

Notation Reference: Secció 4.6.2 [Changing spacing], pàgina 569, Secció 4.6.1 [Displaying spacing], pàgina 568, Secció 4.6 [Fitting music onto fewer pages], pàgina 567, Secció 4.1 [Page layout], pàgina 521, Secció 4.2 [Score layout], pàgina 531, [Separating systems], pàgina 193, Secció 4.2.2 [Setting the staff size], pàgina 533, Secció 4.3 [Breaks], pàgina 535, Secció 4.4 [Vertical spacing], pàgina 541.

Internals Reference: Secció “VerticalAxisGroup” in *Referència de funcionament intern*, Secció “StaffGrouper” in *Referència de funcionament intern*.

Divided voices

Using arpeggioBracket to make divisi more visible

The `arpeggioBracket` can be used to indicate the division of voices where there are no stems to provide the information. This is often seen in choral music.

```
\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 }
}
```



Vegeu també

Notation Reference: Secció 1.3.3 [Expressive marks as lines], pàgina 137.

2.1.6 Opera and stage musicals

The music, lyrics and dialogue to opera and stage musicals are usually set out in one or more of the following forms:

- A *Conductors' Score* containing the full orchestral and vocal parts, together with libretto cues if there are spoken passages.
- *Orchestral Parts* containing the music for the individual instruments of the orchestra or band.
- A *Vocal Score* containing all vocal parts with piano accompaniment. The accompaniment is usually an orchestral reduction, and if so the name of the original orchestral instrument is often indicated. Vocal scores sometimes includes stage directions and libretto cues.
- A *Vocal Book* containing just the vocal parts (no accompaniment), sometimes combined with the libretto.
- A *Libretto* containing the extended passages of spoken dialogue usually found in musicals, together with the words to the sung parts. Stage directions are usually included. LilyPond can be used to typeset libretti but as they contain no music alternative methods may be preferable.

The sections in the LilyPond documentation which cover the topics needed to create scores in the styles commonly found in opera and musicals are indicated in the References below. This is followed by sections covering those techniques which are peculiar to typesetting opera and musical scores.

References for opera and stage musicals

- A conductors' score contains many grouped staves and lyrics. Ways of grouping staves is shown in [Grouping staves], pàgina 187. To nest groups of staves see [Nested staff groups], pàgina 191.
- The printing of empty staves in conductors' scores and vocal scores is often suppressed. To create such a "Frenched score" see [Hiding staves], pàgina 200.
- Writing orchestral parts is covered in Secció 1.6.3 [Writing parts], pàgina 202. Other sections in the Specialist notation chapter may be relevant, depending on the orchestration used. Many instruments are transposing instruments, see [Instrument transpositions], pàgina [undefined].
- If the number of systems per page changes from page to page it is customary to separate the systems with a system separator mark. See [Separating systems], pàgina 193.
- For details of other page formatting properties, see Secció 4.1 [Page layout], pàgina 521.
- Dialogue cues, stage directions and footnotes can be inserted, see Secció 3.2.4 [Creating footnotes], pàgina 483, and Secció 1.8 [Text], pàgina 229. Extensive stage directions can also be added with a section of stand-alone markups between two `\score` blocks, see [Separate text], pàgina 236.

Vegeu també

Musical Glossary: Secció "Frenched score" in *Glossari musical*, Secció "Frenched staves" in *Glossari musical*, Secció "transposing instrument" in *Glossari musical*.

Notation Reference: Secció 3.2.4 [Creating footnotes], pàgina 483, [Grouping staves], pàgina 187, [Hiding staves], pàgina 200, \langle undefined \rangle [Instrument transpositions], pàgina \langle undefined \rangle , [Nested staff groups], pàgina 191, Secció 4.1 [Page layout], pàgina 521, [Separating systems], pàgina 193, \langle undefined \rangle [Transpose], pàgina \langle undefined \rangle , Secció 1.6.3 [Writing parts], pàgina 202, Secció 1.8.1 [Writing text], pàgina 230.

Snippets: Secció “Vocal music” in *Fragments de codi*.

Character names

Character names are usually shown to the left of the staff when the staff is dedicated to that character alone:

```
\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
      }
    }
  >>
}
```

The image shows two musical staves. The top staff is labeled 'KASPAR' and the bottom staff is labeled 'MELCHIOR'. Both staves have a treble clef and a common time signature 'C'. The top staff contains the notes C4, D4, E4, F4, G4. The bottom staff contains the notes C3, D3, E3, F3, G3. A small '8' is written below the first note of the top staff.

When two or more characters share a staff the character’s name is usually printed above the staff at the start of every section applying to that character. This can be done with markup. Often a specific font is used for this purpose.

```
\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
}

```



Alternatively, if there are many character changes, it may be easier to set up variables to hold the definitions for each character so that the switch of characters can be indicated easily and concisely.

```

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
}

```



Vegeu també

Learning Manual: Secció “Organizing pieces with variables” in *Manual d’aprenentatge*.

Notation Reference: Secció 1.8 [Text], pàgina 229, Secció A.11 [Text markup commands], pàgina 674.

Musical cues

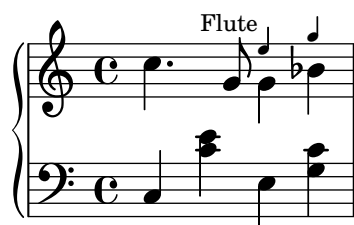
Musical cues can be inserted in Vocal Scores, Vocal Books and Orchestral Parts to indicate what music in another part immediately precedes an entry. Also, cues are often inserted in the piano reduction in Vocal Scores to indicate what each orchestral instrument is playing. This aids the conductor when a full Conductors' Score is not available.

The basic mechanism for inserting cues is fully explained in the main text, see [Quoting other voices], pàgina 206, and [Formatting cue notes], pàgina 209. But when many cues have to be inserted, for example, as an aid to a conductor in a vocal score, the instrument name must be positioned carefully just before and close to the start of the cue notes. The following example shows how this is done.

```
flute = \relative {
  s4 s4 e'' g
}
\addQuote "flute" { \flute }

pianoRH = \relative {
  c''4. g8
  % position name of cue-ing instrument just before the cue notes,
  % and above the staff
  <>^\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
    }
  >>
}
```



If a transposing instrument is being quoted the instrument part should specify its key so the conversion of its cue notes will be done automatically. The example below shows this transposition for a B-flat clarinet. The notes in this example are low on the staff so **DOWN** is specified in `\cueDuring` (so the stems are down) and the instrument name is positioned below the staff.

```
clarinet = \relative c' {
  \transposition bes
  fis4 d d c
}
```

```

\addQuote "clarinet" { \clarinet }

pianoRH = \relative c'' {
  \transposition c'
  % position name of cue-ing instrument below the staff
  <>_\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
      }
    >>
  >>
}

```



From these two examples it is clear that inserting many cues in a Vocal Score would be tedious, and the notes of the piano part would become obscured. However, as the following snippet shows, it is possible to define a music function to reduce the amount of typing and to make the piano notes clearer.

Fragments de codi seleccionats

Adding orchestral cues to a vocal score

This shows one approach to simplify adding many orchestral cues to the piano reduction in a vocal score. The music function `\cueWhile` takes four arguments: the music from which the cue is to be taken, as defined by `\addQuote`, the name to be inserted before the cue notes, then either `#UP` or `#DOWN` to specify either `\voiceOne` with the name above the staff or `\voiceTwo` with the name below the staff, and finally the piano music in parallel with which the cue notes are to appear. The name of the cued instrument is positioned to the left of the cued notes. Many passages can be cued, but they cannot overlap each other in time.

```

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
      }
    >>
  >>
}

```




Vegeu també

Musical Glossary: Secció “cue-notes” in *Glossari musical*.

Notation Reference: Secció 5.5.1 [Aligning objects], pàgina 618, Secció 5.4.2 [Direction and placement], pàgina 602, [Formatting cue notes], pàgina 209, [Quoting other voices], pàgina 206, Secció 5.6 [Using music functions], pàgina 631.

Snippets: Secció “Vocal music” in *Fragments de codi*.

Internals Reference: Secció “CueVoice” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

`\cueDuring` automatically inserts a `CueVoice` context and all cue notes are placed in that context. This means it is not possible to have two overlapping sequences of cue notes by this technique. Overlapping sequences could be entered by explicitly declaring separate `CueVoice` contexts and using `\quoteDuring` to extract and insert the cue notes.

Spoken music

Such effects as ‘parlato’ or ‘Sprechgesang’ require performers to speak without pitch but still with rhythm; these are notated by cross note heads, as demonstrated in `<undefined>` [Special note heads], pàgina `<undefined>`.

Dialogue over music

Dialogue over music is usually printed over the staves in an italic font, with the start of each phrase keyed in to a particular music moment.

For short interjections a simple markup suffices.

```
\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
}
```



For longer phrases it may be necessary to expand the music to make the words fit neatly. There is no provision in LilyPond to do this fully automatically, and some manual intervention to layout the page will be necessary.

For long phrases or for passages with a lot of closely packed dialogue, using a Lyrics context will give better results. The Lyrics context should not be associated with a music Voice; instead each section of dialogue should be given an explicit duration. If there is a gap in the dialogue, the final word should be separated from the rest and the duration split between them so that the underlying music spaces out smoothly.

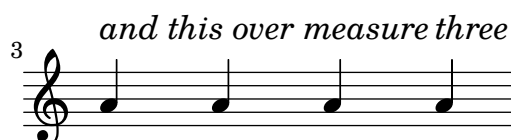
If the dialogue extends for more than one line it will be necessary to manually insert `\breaks` and adjust the placing of the dialogue to avoid running into the right margin. The final word of the last measure on a line should also be separated out, as above.

Here is an example illustrating how this might be done.

```
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 }
    }
  >>
}
```



Vegeu també

Notation Reference: [Manual syllable durations], pàgina 261, Secció 1.8 [Text], pàgina 229.

Internal Reference: Secció “LyricText” in *Referència de funcionament intern*.

2.1.7 Chants psalms and hymns

The music and words for chants, psalms and hymns usually follow a well-established format in any particular church. Although the formats may differ from church to church the type-setting problems which arise are broadly similar, and are covered in this section.

References for chants and psalms

Typesetting Gregorian chant in various styles of ancient notation is described in Secció 2.9 [Ancient notation], pàgina 424.

Vegeu també

Notation reference: Secció 2.9 [Ancient notation], pàgina 424.

Snippets: Secció “Vocal music” in *Fragments de codi*.

Setting a chant

Modern chant settings use modern notation with varying numbers of elements taken from ancient notation. Some of the elements and methods to consider are shown here.

Chants often use quarter notes without stems to indicate the pitch, with the rhythm being taken from the spoken rhythm of the words.

```
stemOff = { \hide Staff.Stem }
```

```
\relative c' {
  \stemOff
  a'4 b c2 |
}
```



Chants often omit the bar lines or use shortened or dotted bar lines to indicate pauses in the music. To omit all bar lines from all staves remove the bar line engraver completely:

```
\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"
    }
  }
}

```



Bar lines can also be removed on a staff-by-staff basis:

```

\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 |
    }
  }
}
>>
}

```



To remove bar lines from just a section of music treat it as a cadenza. If the section is long you may need to insert dummy bar lines with `\bar ""` to show where the line should break.

```

\relative a' {
  a4 b c2 |
  \cadenzaOn

```

```

a4 b c2
a4 b c2
\bar ""
a4 b c2
a4 b c2
\cadenzaOff
a4 b c2 |
a4 b c2 |
}

```



Rests or pauses in chants can be indicated by modified bar lines.

```

\relative a' {
  a4
  \cadenzaOn
  b c2
  a4 b c2
  \bar "'
  a4 b c2
  a4 b c2
  \bar ";
  a4 b c2
  \bar "!"
  a4 b c2
  \bar "||"
}

```



Alternatively, the notation used in Gregorian chant for pauses or rests is sometimes used even though the rest of the notation is modern. This uses a modified `\breathe` mark:

```

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"
    }
  }
}

```



Chants usually omit the time signature and often omit the clef too.

```

\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"
    }
  }
}

```



Chants for psalms in the Anglican tradition are usually either *single*, with 7 bars of music, or *double*, with two lots of 7 bars. Each group of 7 bars is divided into two halves, corresponding

to the two halves of each verse, usually separated by a double bar line. Only whole and half notes are used. The 1st bar in each half always contains a single chord of whole notes. This is the “reciting note”. Chants are usually centered on the page.

```
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 |
}

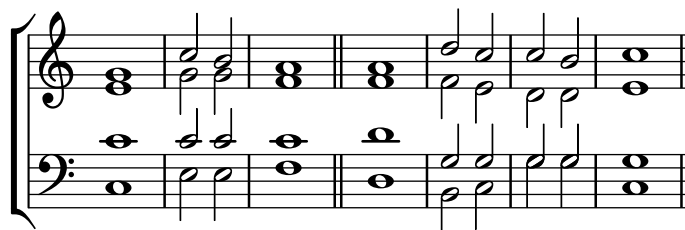
global = {
  \time 2/2
}

% Use markup to center the chant on the page
\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

```



Some other approaches to setting such a chant are shown in the first of the following snippets.

Fragments de codi seleccionats

Chant or psalms notation

This form of notation is used for Psalm chant, where verses aren't always the same length.

```

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 "||"
  }
}

```




Canticles and other liturgical texts may be set more freely, and may use notational elements from ancient music. Often the words are shown underneath and aligned with the notes. If so, the notes are spaced in accordance with the syllables rather than the notes' durations.

Ancient notation template – modern transcription of gregorian music

This example demonstrates how to do modern transcription of Gregorian music. Gregorian music has no measure, no stems; it uses only half and quarter note heads, and special marks, indicating rests of different length.

```
\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
    }
  }
}
```



Vegeu també

Learning Manual: Secció “Visibility and color of objects” in *Manual d’aprenentatge*, Secció “Vocal ensembles templates” in *Manual d’aprenentatge*.

Notation Reference: Secció 2.9 [Ancient notation], pàgina 424, [Bar lines], pàgina 97, Secció 5.1.4 [Modifying context plug-ins], pàgina 579, Secció 2.9.4 [Typesetting Gregorian chant], pàgina 435, [Unmetered music], pàgina 73, Secció 5.4.7 [Visibility of objects], pàgina 610.

Pointing a psalm

The words to an Anglican psalm are usually printed in separate verses centered underneath the chant.

Single chants (with 7 bars) are repeated for every verse. Double chants (with 14 bars) are repeated for every pair of verses. Marks are inserted in the words to show how they should be fitted to the chant. Each verse is divided into two halves. A colon is usually used to indicate this division. This corresponds to the double bar line in the music. The words before the colon are sung to the first three bars of music; the words after the colon are sung to the last four bars.

Single bar lines (or in some psalters an inverted comma or similar symbol) are inserted between words to indicate where the bar lines in the music fall. In markup mode a single bar line can be entered with the bar check symbol, |.

```
\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.

Other symbols may require glyphs from the `fetaMusic` fonts. For details, see Secció 1.8.3 [Fonts], pàgina 251.

```
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.

Where there is one whole note in a bar all the words corresponding to that bar are recited on that one note in speech rhythm. Where there are two notes in a bar there will usually be only one or two corresponding syllables. If there are more than two syllables a dot is usually inserted to indicate where the change in note occurs.

```
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 some psalters an asterisk is used to indicate a break in a recited section instead of a comma, and stressed or slightly lengthened syllables are indicated in bold text.

```
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 other psalters an accent is placed over the syllable to indicate stress.

```

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 rejoíce in the | strength of | our
sal- | -vation.

The use of markup to center text, and arrange lines in columns is described in Secció 1.8.2 [Formatting text], pàgina 237.

Most of these elements are shown in one or other of the two verses in the template, see Secció “Psalms” in *Manual d’aprenentatge*.

Vegeu també

Learning Manual: Secció “Psalms” in *Manual d’aprenentatge*, Secció “Vocal ensembles templates” in *Manual d’aprenentatge*.

Notation Reference: Secció 1.8.3 [Fonts], pàgina 251, Secció 1.8.2 [Formatting text], pàgina 237.

Partial measures in hymn tunes

Hymn tunes frequently start and end every line of music with partial measures so that each line of music corresponds exactly with a line of text. This requires a `\partial` command at the start of the music and `\bar "|"` or `\bar "||"` commands at the end of each line.

Hymn template

This code shows one way of setting out a hymn tune when each line starts and ends with a partial measure. It also shows how to add the verses as stand-alone text under the music.

```
Timeline = {
  \time 4/4
  \tempo 4=96
  \partial 2
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||" \break
  s2 | s1 | s2 \breathe 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

```

The image displays a musical score for the song "The Rose Tree". It consists of two systems of music, each featuring a grand staff with a treble and bass clef. The key signature is one sharp (F#), and the time signature is common time (C). The tempo is marked as quarter note = 96. The melody is written in the treble clef, and the accompaniment is in the bass clef. The first system ends with a repeat sign, and the second system continues the melody and accompaniment.

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 Ancient vocal music

Ancient vocal music is supported, as explained in Secció 2.9 [Ancient notation], pàgina 424.

Vegeu també

Notation Reference: Secció 2.9 [Ancient notation], pàgina 424.

2.2 Keyboard and other multi-staff instruments

Un peu retenu
très expressif

ppp

Rall. long

a Tempo

pp

ped.



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], pàgina 173.
- Keyboard music can be written in parallel, as described in [Writing music in parallel], pàgina 183.
- 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], pàgina 122.
- Fingerings are indicated with [Fingering instructions], pàgina 219.
- Organ pedal indications are inserted as articulations, see Secció A.14 [List of articulations], pàgina 732.
- Vertical grid lines can be shown with [Grid lines], pàgina 226.
- Keyboard music often contains *Laissez vibrer* ties as well as ties on arpeggios and tremolos, described in [Ties], pàgina 53.
- Placing arpeggios across multiple voices and staves is covered in [Arpeggio], pàgina 142.
- Tremolo marks are described in [Tremolo repeats], pàgina 162.
- Several of the tweaks that can occur in keyboard music are demonstrated in Secció “Real music example” in *Manual d’aprenentatge*.
- Hidden notes can be used to produce ties that cross voices, as shown in Secció “Other uses for tweaks” in *Manual d’aprenentatge*.

Vegeu també

Learning Manual: Secció “Real music example” in *Manual d’aprenentatge*, Secció “Other uses for tweaks” in *Manual d’aprenentatge*.

Notation Reference: [Grouping staves], pàgina 187, [Instrument names], pàgina 202, [Collision resolution], pàgina 173, [Writing music in parallel], pàgina 183, [Fingering instructions], pàgina 219, Secció A.14 [List of articulations], pàgina 732, [Grid lines], pàgina 226, [Ties], pàgina 53, [Arpeggio], pàgina 142, [Tremolo repeats], pàgina 162.

Internals Reference: Secció “PianoStaff” in *Referència de funcionament intern*.

Snippets: Secció “Keyboards” in *Fragments de codi*.

Changing staff manually

Voices can be switched between staves manually, using the command

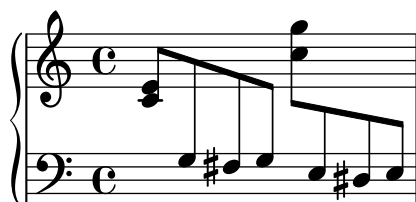
```
\change Staff = staffname
```

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 Secció 5.1.3 [Keeping contexts alive], pàgina 576.

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 Secció “Fixing overlapping notation” in *Manual d’aprenentatge*.

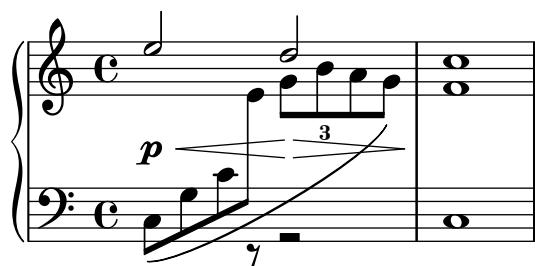
Overlapping notation can result when voices cross staves:

```
\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 Secció “Fixing overlapping notation” in *Manual d’aprenentatge*.

Vegeu també

Learning Manual: Secció “Fixing overlapping notation” in *Manual d’aprenentatge*.

Notation Reference: [Stems], pàgina 224, [Automatic beams], pàgina 82, Secció 5.1.3 [Keeping contexts alive], pàgina 576.

Snippets: Secció “Keyboards” in *Fragments de codi*.

Internals Reference: Secció “Beam” in *Referència de funcionament intern*, Secció “ContextChange” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

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.

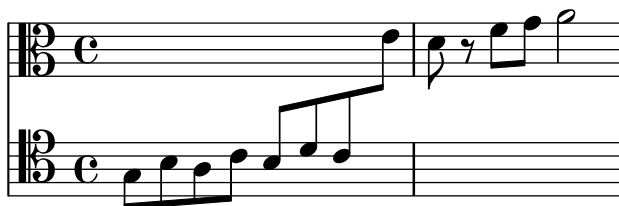
```
\new PianoStaff {
  \autochange {
    g4 a b c'
    d'4 r a g
  }
}
```



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
```



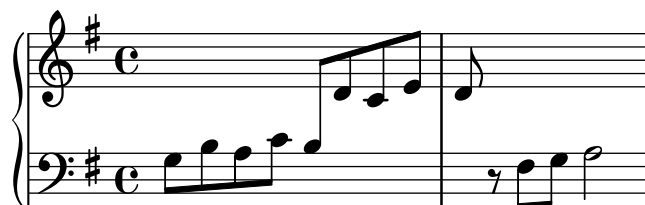
A `\relative` section that is outside of `\autochange` has no effect on the pitches of the music, so if necessary, put `\relative` inside `\autochange`.

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
  }
>>
```



Vegeu també

Notation Reference: [Changing staff manually], pàgina 319.

Snippets: Secció “Keyboards” in *Fragments de codi*.

Internals Reference: Secció “AutoChangeMusic” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

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

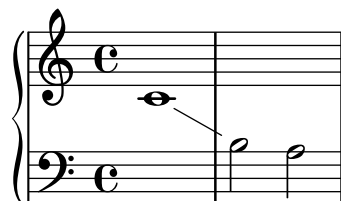
Whenever a voice switches to another staff, a line connecting the notes can be printed automatically:

```
\new PianoStaff <<
  \new Staff = "one" {
    \showStaffSwitch
    c'1
    \change Staff = "two"
```

```

    b2 a
  }
  \new Staff = "two" {
    \clef bass
    s1*2
  }
>>

```



Instruccions predefinides

`\showStaffSwitch`, `\hideStaffSwitch`.

Vegeu també

Snippets: Secció “Keyboards” in *Fragments de codi*.

Internals Reference: Secció “Note_head_line_engraver” in *Referència de funcionament intern*,
Secció “VoiceFollower” in *Referència de funcionament intern*.

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.

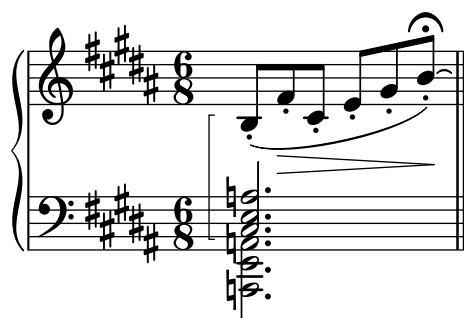
Fragments de codi seleccionats

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.
        }
      >>
    }
  }
>>
```



Vegeu també

Snippets: Secció “Keyboards” in *Fragments de codi*.

Internals Reference: Secció “Stem” in *Referència de funcionament intern*.

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.

Vegeu també

Notation Reference: [Ties], pàgina 53.

Snippets: Secció “Keyboards” in *Fragments de codi*.

Internals Reference: Secció “SustainPedal” in *Referència de funcionament intern*, Secció “SustainPedalLineSpanner” in *Referència de funcionament intern*, Secció “SustainEvent” in *Referència de funcionament intern*, Secció “SostenutoPedal” in *Referència de funcionament intern*, Secció “SostenutoPedalLineSpanner” in *Referència de funcionament intern*, Secció “SostenutoEvent” in *Referència de funcionament intern*, Secció “UnaCordaPedal” in *Referència de funcionament intern*, Secció “UnaCordaPedalLineSpanner” in *Referència de funcionament intern*, Secció “UnaCordaEvent” in *Referència de funcionament intern*, Secció “PianoPedalBracket” in *Referència de funcionament intern*, Secció “Piano-pedal-engraver” in *Referència de funcionament intern*.

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.

Fragments de codi seleccionats

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
```

```
     \clef bass \stdBass "Master"
```

```
     <<
```

```
       { r16 <f, bes d>^"b" r <e a c>^"am" r <d g bes>^"gm" |
```

```
       <e a cis>1^"a" }
```

```
       \\\
```

```
       { d8_"D" c_"C" bes_"B" | a1_"A" }
```

```
     >>
```




Vegeu també

Snippets: Secció “Keyboards” in *Fragments de codi*.

2.2.4 Harp

This section discusses notation issues that are unique to the harp.

References for harps

Some common characteristics of harp music are covered elsewhere:

- The glissando is the most characteristic harp technique, [Glissando], pàgina 137.
- A *bisbigliando* is written as a tremelo [Tremolo repeats], pàgina 162.
- Natural harmonics are covered under [Harmonics], pàgina 330.
- For directional arpeggios and non-arpeggios, see [Arpeggio], pàgina 142.

Vegeu també

Notation Reference: [Tremolo repeats], pàgina 162, [Glissando], pàgina 137, [Arpeggio], pàgina 142, [Harmonics], pàgina 330.

Harp pedals

Harps have seven strings per octave that may be sounded at the natural, flattened, or sharpened pitch. In lever harps, each string is adjusted individually, but in pedal harps every string with the same pitch name is controlled by a single pedal. From the player’s left to right, the pedals are D, C, and B on the left and E, F, G, and A on the right. The position of the pedals may be indicated with text marks:

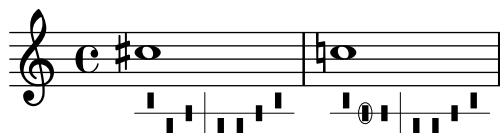
```
\textLengthOn
cis''1_markup \concat \vcenter {
  [D \flat C \sharp B|E \sharp F \sharp G A \flat] }
c''!1_markup \concat \vcenter {
  [ C \natural ] }
```



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.

Vegeu també

Notation Reference: [Text scripts], pàgina 230, Secció A.11.5 [Instrument Specific Markup], pàgina 713.

2.3 Unfretted string instruments

lentement

1 *fatigué* s. vib. 1) n. 2) s.p. 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.

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], pàgina 230.
- Fingerings, including the thumb indication, are described in [Fingering instructions], pàgina 219.
- Double stopping is normally indicated by writing a chord, see [Chorded notes], pàgina 164. Directives for playing chords may be added, see [Arpeggio], pàgina 142.
- Templates for string quartets can be found in Secció “String quartet templates” in *Manual d’aprenentatge*. Others are shown in the snippets.

Vegeu també

Learning Manual: Secció “String quartet templates” in *Manual d’aprenentatge*.

Notation Reference: [Text scripts], pàgina 230, [Fingering instructions], pàgina 219, [Chorded notes], pàgina 164, [Arpeggio], pàgina 142.

Snippets: Secció “Unfretted strings” in *Fragments de codi*.

Bowing indications

Bowing indications are created as articulations, which are described in [Articulations and ornamentations], pàgina 119.

The bowing commands, `\upbow` and `\downbow`, are used with slurs as follows:

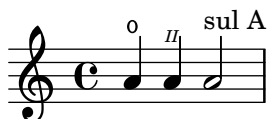
```
\relative { c'4(\downbow d) e(\upbow f) }
```



Roman numerals can be added as strings numbers (rather than the default circled Arabic numbers), as explained in [String number indications], pàgina 332.

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" }
```



Instruccions predefinides

`\downbow`, `\upbow`, `\open`, `\romanStringNumbers`.

Vegeu també

Notation Reference: [Articulations and ornamentations], pàgina 119, [String number indications], pàgina 332, [Slurs], pàgina 130.

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], pàgina 329.

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
}
```



Vegeu també

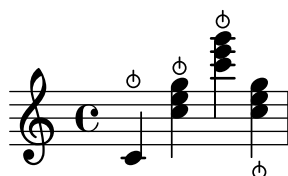
Music Glossary: Secció “harmonics” in *Glossari musical*.

Notation Reference: `<undefined>` [Special note heads], pàgina `<undefined>`, [References for unfretted strings], pàgina 329.

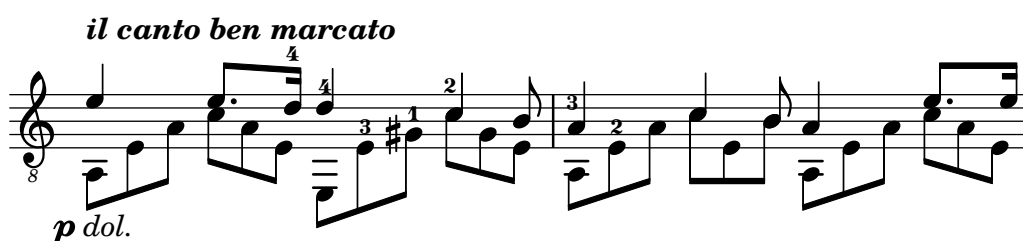
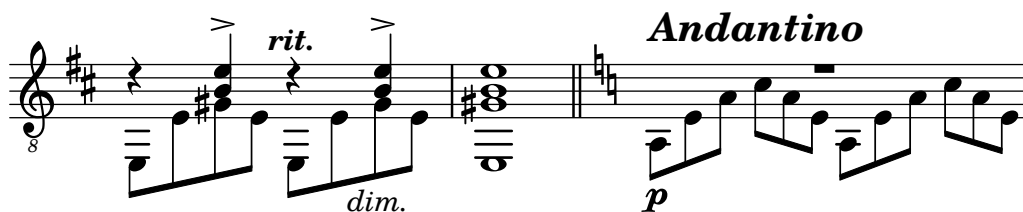
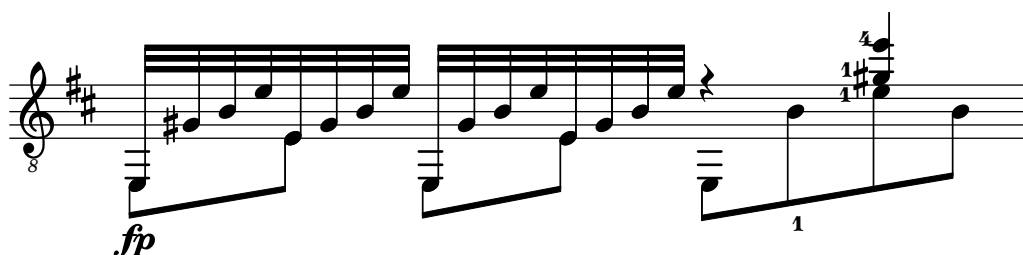
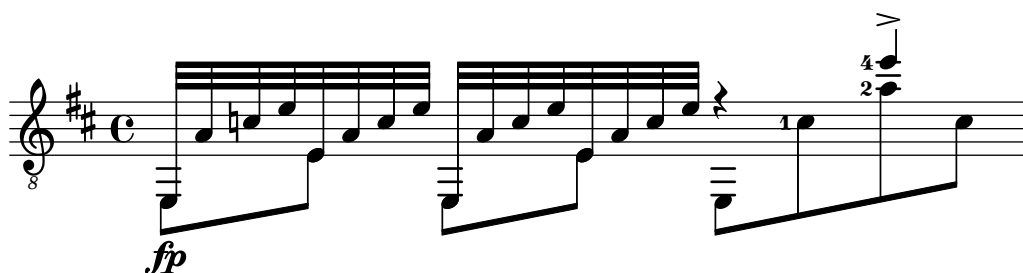
Snap (Bartók) pizzicato

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





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], pàgina 219.
- Instructions for *Laissez vibrer* ties as well as ties on arpeggios and tremolos can be found in [Ties], pàgina 53.
- Instructions for handling multiple voices can be found in [Collision resolution], pàgina 173.
- Instructions for indicating harmonics can be found in [Harmonics], pàgina 330.

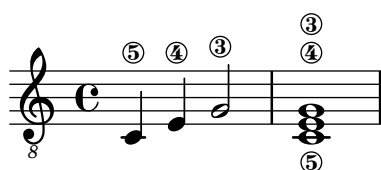
Vegeu també

Notation Reference: [Fingering instructions], pàgina 219, [Ties], pàgina 53, [Collision resolution], pàgina 173, [Instrument names], pàgina 202, [Writing music in parallel], pàgina 183, [Arpeggio], pàgina 142, Secció A.14 [List of articulations], pàgina 732, <undefined> [Clef], pàgina <undefined>, <undefined> [Instrument transpositions], pàgina <undefined>.

String number indications

The string on which a note should be played may be indicated by appending `\number` to a note.

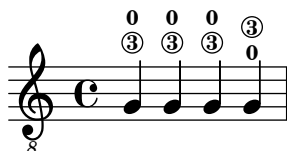
```
\clef "treble_8"
c4\5 e\4 g2\3
<c\5 e\4 g\3>1
```



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
```

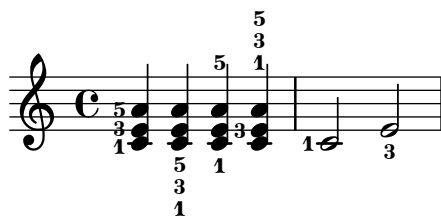


Fragments de codi seleccionats

Controlling the placement of chord fingerings

The placement of fingering numbers can be controlled precisely. For fingering orientation to apply, you must use a chord construct <> even if it is a single note.

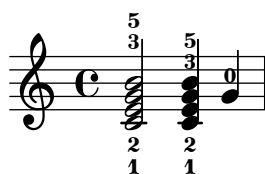
```
\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down right up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(left)
  <c-1>2
  \set fingeringOrientations = #'(down)
  <e-3>2
}
```



Allowing fingerings to be printed inside the staff

By default, vertically oriented fingerings are positioned outside the staff. However, this behavior can be canceled. Note: you must use a chord construct `<>`, even if it is only a single note.

```
\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>
}
```



Instruccions predefinides

`\arabicStringNumbers`, `\romanStringNumbers`.

Vegeu també

Notation Reference: [Fingering instructions], pàgina 219.

Snippets: Secció “Fretted strings” in *Fragments de codi*.

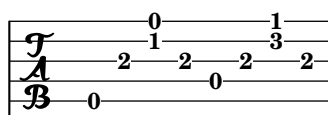
Internals Reference: Secció “StringNumber” in *Referència de funcionament intern*, Secció “Fingering” in *Referència de funcionament intern*.

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 expressive marks, for example.

```
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

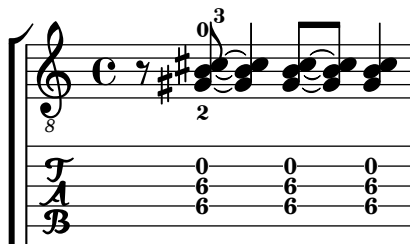
```
\chordRepeats #'(string-number-event fingering-event)
```

explicitly on music expressions in tabulature using [Chord repetition], pàgina 166. 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
```

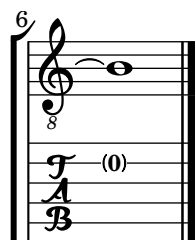
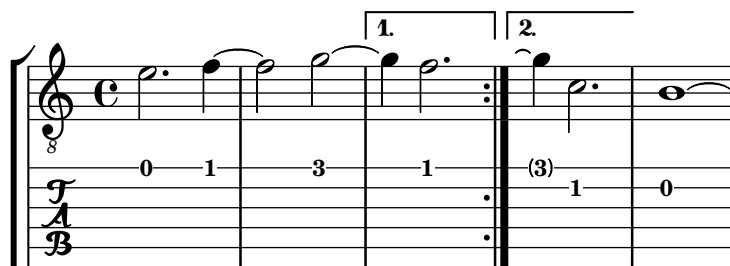
```
}
>>
```



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 "|."
}

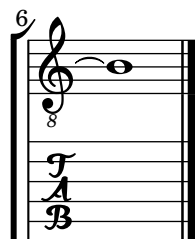
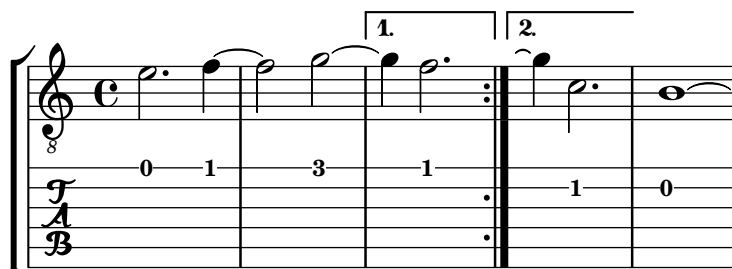
\score {
  <<
    \new StaffGroup <<
      \new Staff {
        \clef "treble_8"
        \ties
      }
      \new TabStaff {
        \ties
      }
    >>
  >>
  \layout {
    indent = #0
    ragged-right = ##t
  }
}
```



The command `\hideSplitTiedTabNotes` cancels the behavior of engraving fret numbers in parentheses:

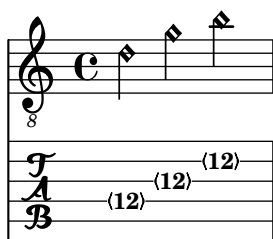
```
ties = \relative {
  \repeat volta 2 {
    e'2. f4~
    2 g2~ }
  \alternative {
    { g4 f2. }
    { g4\repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}

\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 }
  >>
}
```



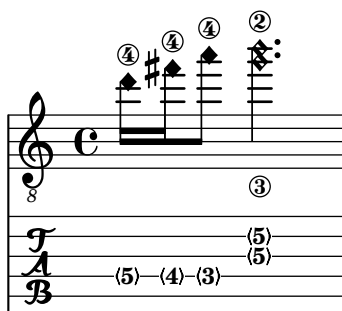
Note that the command `\harmonic` must always be attached to single notes (possibly inside of a chord) instead of whole chords. It only makes sense for open-string harmonics in the 12th fret. All other harmonics should be calculated by LilyPond. This can be achieved by indicating the fret where a finger 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 }
>>
}

```

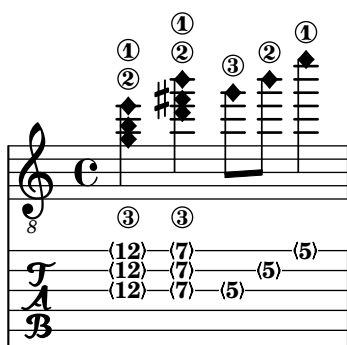


Alternatively, harmonics can be computed by defining the ratio of string lengths above and below the harmonic fingering.

```

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 }
  >>
}

```



Fragments de codi seleccionats

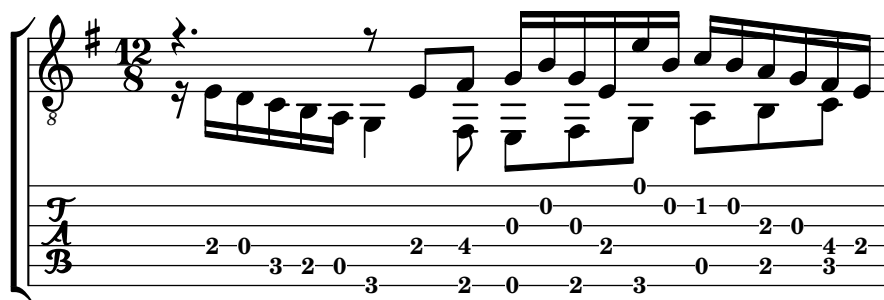
Stem and beam behavior in tablature

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 {

```

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)

4th harm. 5th harm.

(4) (4) (9) (9) (16) (16) (3) (3)

10

8 6th harm. ... 7th harm. ... 8th harm. ...

(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
      }
    }
  >>
}

```

The image displays a musical score with two staves. The top staff is a treble clef staff with a common time signature. It contains a series of notes, each marked with a diamond symbol. Below the notes are labels: AH 16, AH 17, AH 19, PH....., TH 17, TH 19, TH 17, and TCH. The bottom staff is a tablature staff with three lines. It shows fret numbers in parentheses: (4), (5), (7), (7), (5), (7), (5), and 2-(9).

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
}

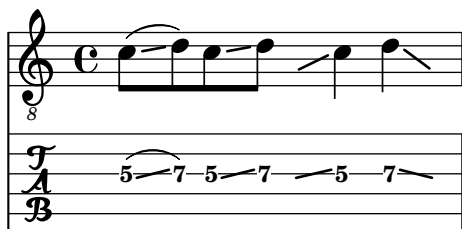
\score {
  <<
    \new Staff { \clef "treble_8" \slides }
    \new TabStaff { \slides }
  >>
}

```

```

\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.

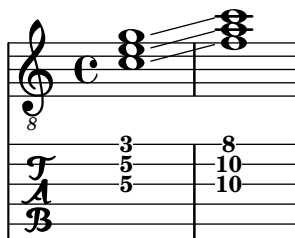
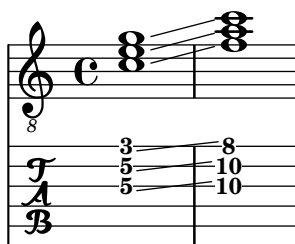
```

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
    }
  >>
}

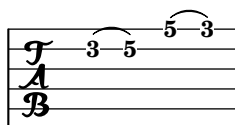
```



Hammer on and pull off

Hammer-on and pull-off can be obtained using slurs.

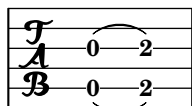
```
\new TabStaff {
  \relative c' {
    d4( e\2)
    a( g)
  }
}
```



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
  }
}
```

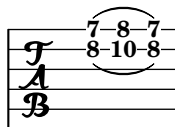


Hammer on and pull off using chords

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>)
  }
}
```

}



Vegeu també

Notation Reference: [Chord repetition], pàgina 166, [Glissando], pàgina 137, [Harmonics], pàgina 330, [Stems], pàgina 224, [Written-out repeats], pàgina 157.

Snippets: Secció “Fretted strings” in *Fragments de codi*.

Internals Reference: Secció “TabNoteHead” in *Referència de funcionament intern*, Secció “TabStaff” in *Referència de funcionament intern*, Secció “TabVoice” in *Referència de funcionament intern*, Secció “Beam” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

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:

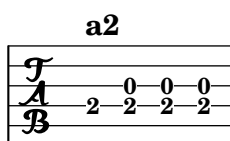
```
melodia = \partcombine { e4 g g g } { e4 e e e }
```

```
<<
```

```
\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.

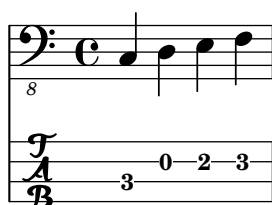
```
<<
```

```
\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.

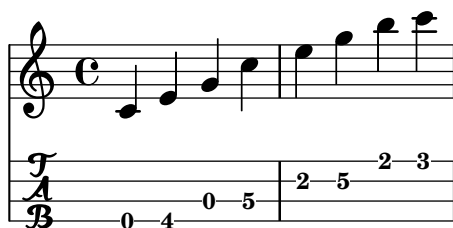
Its argument is a chord construct defining the pitches of each string in the tuning. The chord construct must be in absolute octave mode, see [\[Absolute octave entry\]](#), pàgina [\[undefined\]](#). The string with the highest number (generally the lowest string) must come first in the chord. For example, 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.

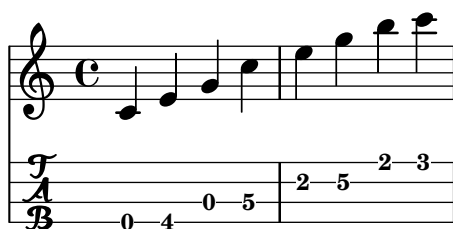
String tunings are used as part of the hash key for predefined fret diagrams (see [Predefined fret diagrams], pàgina 362).

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 Secció A.22 [Scheme functions], pàgina 786).

`\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

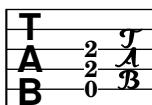
```
\layout {
  \context {
    \TabStaff
    stringTunings = \stringTuning <c' g' d'' a''>
  }
}
```



```
}
```

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
  eis
  eisih
  eisis
}
```

```
<<
  \new Staff << \clef "G_8" \mus >>
  \new TabStaff \with { stringTunings = \custom-tuning } \mus
>>
```

The musical score for 'The Rose Tree' is presented in two systems. The first system consists of a treble clef staff with a common time signature 'C' and a key signature of one flat (B-flat). The melody is written in eighth notes. Below the staff is a bass line with a treble clef and a series of numbers: 3 1/2, 4, 4 1/2, 0, 1/2, 1, 1 1/2, 2, 2 1/2. The second system is a continuation of the melody on a new treble clef staff, also in common time and one flat key signature. The bass line continues with the same sequence of numbers.

Vegeu també

Notation Reference: [\[Absolute octave entry\]](#), [pàgina \[\\[Predefined fret diagrams\\]\]\(#\)](#), [pàgina 362](#), [Secció A.22](#) [\[Scheme functions\]](#), [pàgina 786](#).

Installed Files: ly/string-tunings-init.ly, scm/tablature.scm.

Snippets: Secció “Fretted strings” in *Fragments de codi*.

Internals Reference: Secció “Tab_note_heads_engraver” in *Referència de funcionament intern*.

Advertisements i problemes coneguts

Automatic tablature calculations do not work properly in most cases for instruments where string pitches do not vary monotonically with string number, such as ukuleles.

Fret diagram markups

Fret diagrams can be added to music as a markup to the desired note. The markup contains information about the desired fret diagram. There are three different fret-diagram markup interfaces: standard, terse, and verbose. The three interfaces produce equivalent markups, but have varying amounts of information in the markup string. Details about the syntax of the different markup strings used to define fret diagrams are found at Secció A.11.5 [Instrument Specific Markup], pàgina 713.

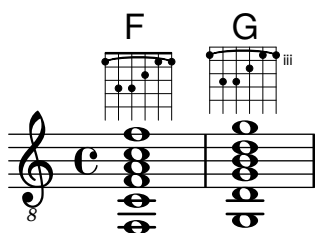
The standard fret diagram markup string indicates the string number and the fret number for each dot to be placed on the string. In addition, open and unplayed (muted) strings can be indicated.

```
<<
\new ChordNames {
  \chordmode {
    c1 d:m
  }
}
\new Staff {
  \clef "treble_8"
  <c e g c' e'>1^\markup {
    \fret-diagram #"6-x;5-3;4-2;3-o;2-1;1-o;"
  }
  <d a d' f'>1^\markup {
    \fret-diagram #"6-x;5-x;4-o;3-2;2-3;1-1;"
  }
}
>>
```

The diagram illustrates the fretboard positions for the C major and Dm minor chords. Above the fretboard, the chord names 'C' and 'Dm' are displayed. The C major chord is shown with an open string (X) on the 1st fret, and open strings (O) on the 2nd, 3rd, and 4th frets. The Dm minor chord is shown with an open string (X) on the 1st fret, and open strings (O) on the 2nd, 3rd, and 4th frets. Below the fretboard, the musical notation for these chords is shown on a staff with a treble clef and a common time signature (C). The C major chord is represented by a whole note chord with notes C4, E4, and G4. The Dm minor chord is represented by a whole note chord with notes D4, F4, and A4.

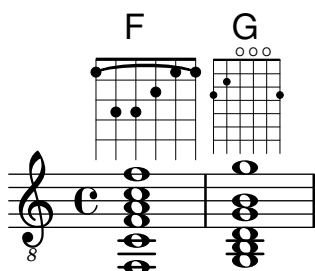
Barre indications can be added to the diagram from the fret-diagram markup string.

```
<<
\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;"
  }
}
>>
```



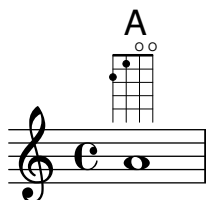
The size of the fret diagram, and the number of frets in the diagram can be changed in the fret-diagram markup string.

```
<<
\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;"
  }
}
>>
```



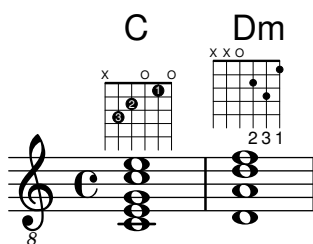
The number of strings in a fret diagram can be changed to accommodate different instruments such as banjos and ukuleles with the fret-diagram markup string.

```
<<
\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;"
  }
}
>>
```



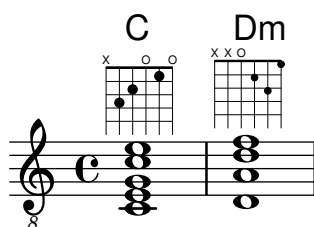
Dot radius and dot position can be controlled with the fret-diagram markup string.

```
<<
```

```

\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;"
  }
}
>>

```

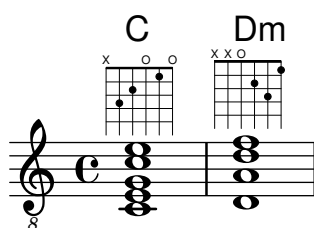


The fret-diagram-terse markup string omits string numbers; the string number is implied by the presence of semicolons. There is one semicolon for each string in the diagram. The first semicolon corresponds to the highest string number and the last semicolon corresponds to the first string. Mute strings, open strings, and fret numbers can be indicated.

```

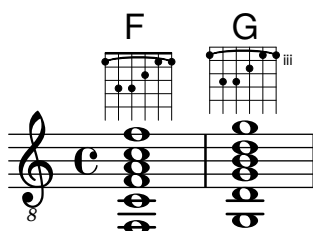
<<
\new ChordNames {
  \chordmode {
    c1 d:m
  }
}
\new Staff {
  \clef "treble_8"
  <c e g c' e'>1^\markup {
    \fret-diagram-terse #"x;3;2;o;1;o;"
  }
  <d a d' f'>1^\markup {
    \fret-diagram-terse #"x;x;o;2;3;1;"
  }
}
>>

```



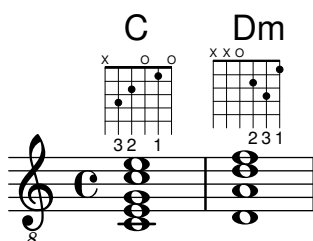
Barre indicators can be included in the fret-diagram-terse markup string.

```
<<
\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.

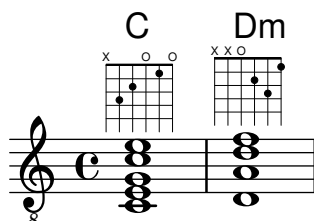
```
<<
\new ChordNames {
  \chordmode {
    c1 d:m
  }
}
\new Staff {
  \override Voice.TextScript.fret-diagram-details.finger-code = #'below-string
  \clef "treble_8"
  <c e g c' e'>1^\markup {
    \fret-diagram-terse #"x;3-3;2-2;o;1-1;o;"
  }
  <d a d' f'>1^\markup {
    \fret-diagram-terse #"x;x;o;2-2;3-3;1-1;"
  }
}
>>
```



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 #`

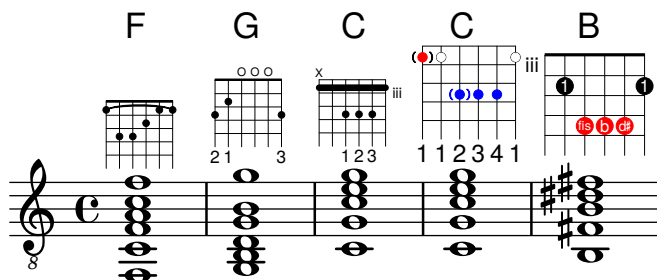
```



```

        (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 Secció “fret-diagram-interface” in *Referència de funcionament intern*. For a fret diagram markup, the interface properties belong to `Voice.TextScript`.

Fragments de codi seleccionats

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"
```

```

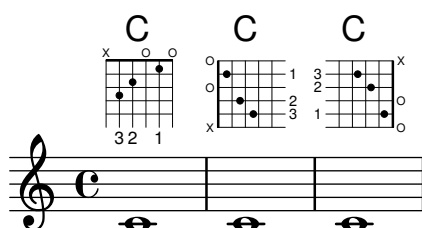
<<
\chords {
  c1
  c1
  c1
}
\new FretBoards {
  \chordmode {
    c1
    \override FretBoard.fret-diagram-details.orientation =
      #'landscape
    c1

```

```

\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)

```

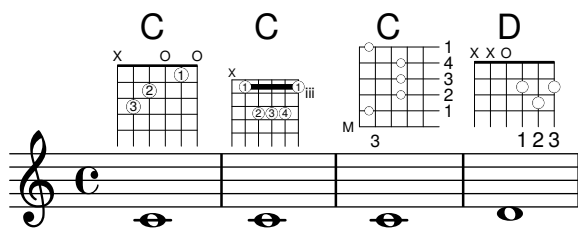
```

        (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;"
  }
}
}
}
>>

```



Vegeu també

Notation Reference: Secció A.11.5 [Instrument Specific Markup], pàgina 713.

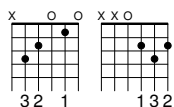
Snippets: Secció “Fretted strings” in *Fragments de codi*.

Internals Reference: Secció “fret-diagram-interface” in *Referència de funcionament intern*.

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`.

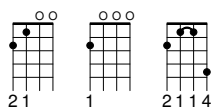
```
\include "predefined-ukulele-fretboards.ly"
```

```
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`.

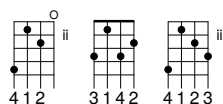
```
\include "predefined-mandolin-fretboards.ly"

myChords = \chordmode { c1 c:m7.5- c:aug }

\new ChordNames {
  \myChords
}

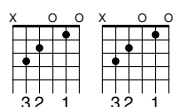
\new FretBoards {
  \set Staff.stringTunings = #mandolin-tuning
  \myChords
}
```

C C[∅] C+



Chord pitches can be entered either as simultaneous music or using chord mode (see [Chord mode overview], pàgina 404).

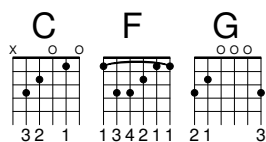
```
\include "predefined-guitar-fretboards.ly"
\new FretBoards {
  \chordmode { c1 }
  <c' e' g'>1
}
```



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
}

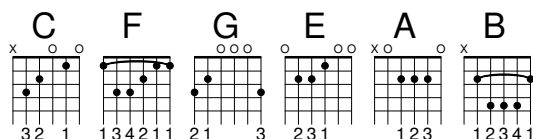
<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



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
}

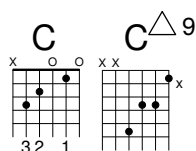
mychordlist = {
  \mychords
  \transpose c e { \mychords }
}
<<
  \new ChordNames {
    \mychordlist
  }
  \new FretBoards {
    \mychordlist
  }
>>
```



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 Secció A.4 [Predefined fretboard diagrams], pàgina 640. 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], pàgina 372.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode{
  c1 c:maj9
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



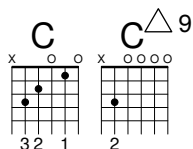
Fret diagrams can be added to the fret diagram table. To add a diagram, you must specify the hash table for the diagram, the chord for the diagram, the tuning to be used, and a definition for the diagram. Normally, the hash table will be *default-fret-table*. The diagram definition can be either a fret-diagram-terse 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
    }
>>
```



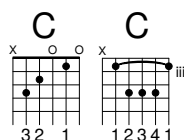
Different fret diagrams for the same chord name can be stored using different octaves of pitches. The different octave should be at least two octaves above or below the default octave, because the octaves above and below the default octave are used for transposing fretboards.

```
\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

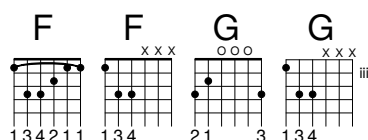
\addChordShape #'powerf #guitar-tuning #"1-1;3-3;3-4;x;x;x;"

% 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 Secció “fret-diagram-interface” in *Referència de funcionament intern*. For a predefined fret diagram, the interface properties belong to `FretBoards.FretBoard`.

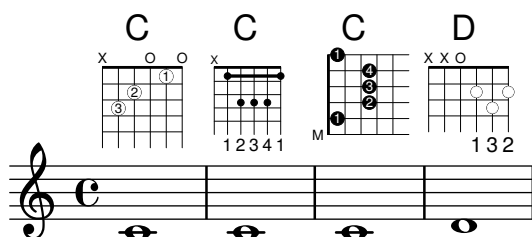
Fragments de codi seleccionats

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 ⁶	D	A ^Δ	D ^Δ	G
T	0	3	2	1	0
A	0	0	1	0	1
B	0	0	2	0	0
	0	0	0	0	2

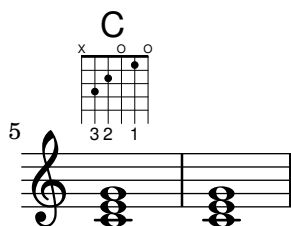
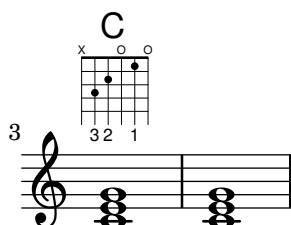
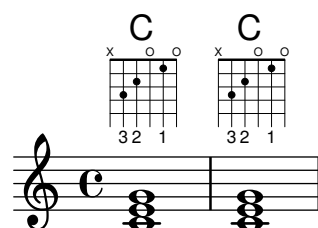
ChordChanges for FretBoards

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 |
      }
    >>
  }
>>

```

Default table New table from empty New table from default

Vegeu també

Notation Reference: [Custom tablatures], pàgina 348, [Automatic fret diagrams], pàgina 372, [Chord mode overview], pàgina 404, Secció A.4 [Predefined fretboard diagrams], pàgina 640.

Installed Files: `ly/predefined-guitar-fretboards.ly`,
`ly/predefined-guitar-ninth-fretboards.ly`,
`ly/predefined-ukulele-fretboards.ly`,
`ly/predefined-mandolin-fretboards.ly`.

Snippets: Secció “Fretted strings” in *Fragments de codi*.

Internals Reference: Secció “fret-diagram-interface” in *Referència de funcionament intern*.

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.

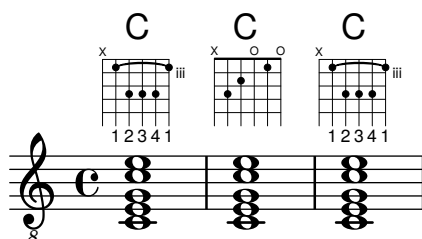
```
<<
\new ChordNames {
  \chordmode {
    f1 g
  }
}
\new FretBoards {
  <f, c f a c' f'>1
  <g,\6 b, d g b g'>1
}
\new Staff {
  \clef "treble_8"
  <f, c f a c' f'>1
  <g, b, d g b' g'>1
}
>>
```

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.

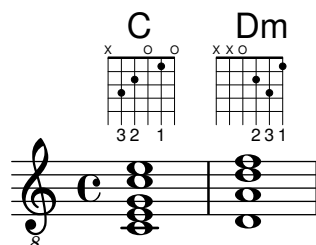
Fingerings can be added to `FretBoard` fret diagrams.

```

<<
  \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
  }
>>

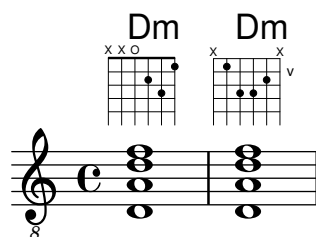
```

```
}
>>
```



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], pàgina 348, 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 Secció “fret-diagram-interface” in *Referència de funcionament intern*. For a `FretBoards` fret diagram, the interface properties belong to `FretBoards.FretBoard`.

Instruccions predefinides

```
\predefinedFretboardsOff, \predefinedFretboardsOn.
```


Vegeu també

Notation Reference: [Custom tablatures], pàgina 348.

Snippets: Secció “Fretted strings” in *Fragments de codi*.

Internals Reference: Secció “fret-diagram-interface” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

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`

Fragments de codi seleccionats

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

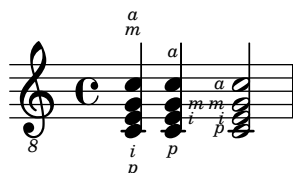
```
#(define RH rightHandFinger)

\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
}
```

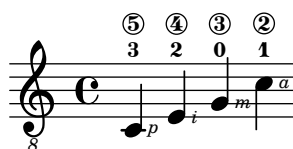


Fingerings string indications and right-hand fingerings

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
}
```



Vegeu també

Snippets: Secció “Fretted strings” in *Fragments de codi*.

Internals Reference: Secció “StrokeFinger” in *Referència de funcionament intern*.

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
}
```



Vegeu també

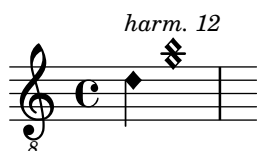
Notation Reference: [Text spanners], pàgina 231.

Snippets: Secció “Fretted strings” in *Fragments de codi*, Secció “Expressive marks” in *Fragments de codi*.

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
}
}

```



Vegeu també

Snippets: Secció “Fretted strings” in *Fragments de codi*.

Notation Reference: [\[Special note heads\]](#), pàgina [\[undefined\]](#), Secció A.9 [\[Note head styles\]](#), pàgina 672.

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 shows a musical score for a guitar. The top staff is a treble clef with a common time signature 'C'. It contains six measures of power chords: E⁵, A⁵, C⁵, F⁵, A⁵, and G⁵. Below the staff is a six-string guitar tablature with six lines labeled T (Treble), A (Acoustic), and B (Bass). The fret numbers for each measure are: Measure 1 (E⁵): T=8, A=2, B=0; Measure 2 (A⁵): T=2, A=2, B=0; Measure 3 (C⁵): T=0, A=0, B=8; Measure 4 (F⁵): T=10, A=10, B=8; Measure 5 (A⁵): T=7, A=7, B=5; Measure 6 (G⁵): T=8, A=7, B=5.

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
  }
  >>
}

```

The image shows a musical score for a guitar. The top staff is a treble clef with a common time signature 'C'. It contains four measures of chords: C, B⁵, F^{#5}, and G^m. Below the staff is a six-string guitar tablature with six lines labeled T (Treble), A (Acoustic), and B (Bass). The fret numbers for each measure are: Measure 1 (C): T=0, A=2, B=3; Measure 2 (B⁵): T=4, A=4, B=2; Measure 3 (F^{#5}): T=4, A=4, B=2; Measure 4 (G^m): T=0, A=1, B=3.

Vegeu també

Music Glossary: Secció “power chord” in *Glossari musical*.

Notation Reference: [Extended and altered chords], pàgina 407, [Printing chord names], pàgina 409.

Snippets: Secció “Fretted strings” in *Fragments de codi*.

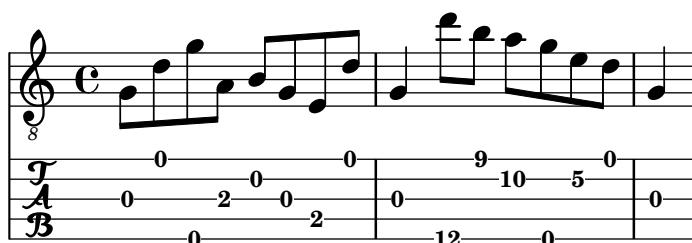
2.4.3 Banjo

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 }
>>
```



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)
```

Vegeu també

Installed Files: `ly/string-tunings-init.ly`.

Snippets: Secció “Fretted strings” in *Fragments de codi*.

2.4.4 Lute

Lute tablatures

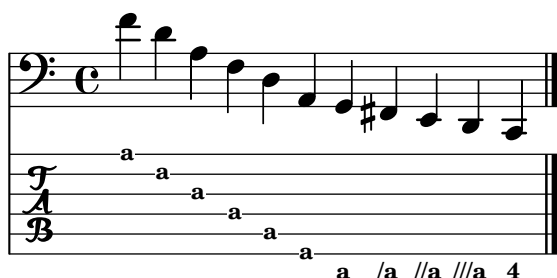
LilyPond supports tablature for lute.

To get additional bass strings use `additionalBassStrings`, where the pitches of those strings are set. They will be printed below lowest line as: a, /a, //a, ///a, 4, 5, etc.

`fret-letter-tablature-format` for `tablatureFormat` should be used, probably `fretLabels` for further customizing.

```
m = { f'4 d' a f d a, g, fis, e, d, c, \bar "|" } }
```

```
\score {
  <<
    \new Staff { \clef bass \cadenzaOn \m }
    \new TabStaff \m
  >>
  \layout {
    \context {
      \Score
      tablatureFormat = #fret-letter-tablature-format
    }
    \context {
      \TabStaff
      stringTunings = \stringTuning <a, d f a d' f'>
      additionalBassStrings = \stringTuning <c, d, e, fis, g,>
      fretLabels = #'("a" "b" "r" "d" "e" "f" "g" "h" "i" "k")
    }
  }
}
```



Advertiments i problemes coneguts

Using `FretBoards` with `additionalBassStrings` is not supported and will yield unsatisfying results.

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], pàgina 79, and [Instantiating new staves], pàgina 186.
- MIDI output is discussed in a separate section; please see Secció A.6 [MIDI instruments], pàgina 649.

Vegeu també

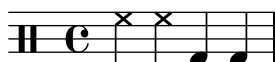
Notation Reference: [Showing melody rhythms], pàgina 79, [Instantiating new staves], pàgina 186. Secció A.6 [MIDI instruments], pàgina 649.

Snippets: Secció “Percussion” in *Fragments de codi*.

Basic percussion notation

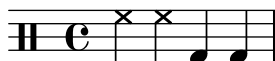
Percussion notes may be entered in `\drummode` mode, which is similar to the standard mode for entering notes. The simplest way to enter percussion notes is to use the `\drums` command, which creates the correct context and entry mode for percussion:

```
\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 Secció A.15 [Percussion notes], pàgina 734.

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 Secció A.6 [MIDI instruments], pàgina 649.

Vegeu també

Notation Reference: Secció A.6 [MIDI instruments], pàgina 649, Secció A.15 [Percussion notes], pàgina 734.

Installed Files: `ly/drumpitch-init.ly`.

Snippets: Secció “Percussion” in *Fragments de codi*.

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], pàgina 162.

```
\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 Secció 5.4.2 [Direction and placement], pàgina 602. 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"
  }
}
```



Vegeu també

Notation Reference: [Tremolo repeats], pàgina 162.

Snippets: Secció “Percussion” in *Fragments de codi*.

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.

Vegeu també

Notation Reference: Secció A.6 [MIDI instruments], pàgina 649.

Snippets: Secció “Percussion” in *Fragments de codi*.

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 }
>>

```

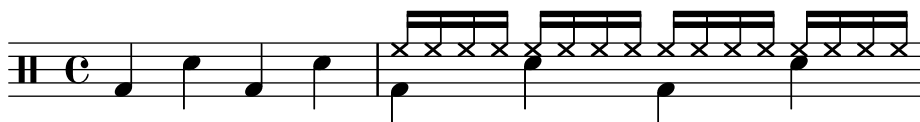


The above example shows verbose polyphonic notation. The short polyphonic notation, described in Secció “I’m hearing Voices” in *Manual d’aprenentatge*, can also be used. For example,

```

\new DrumStaff <<
    \drummode {
        bd4 sn4 bd4 sn4
        << {
            \repeat unfold 16 hh16
        } \\ {
            bd4 sn4 bd4 sn4
        } >>
    }
>>

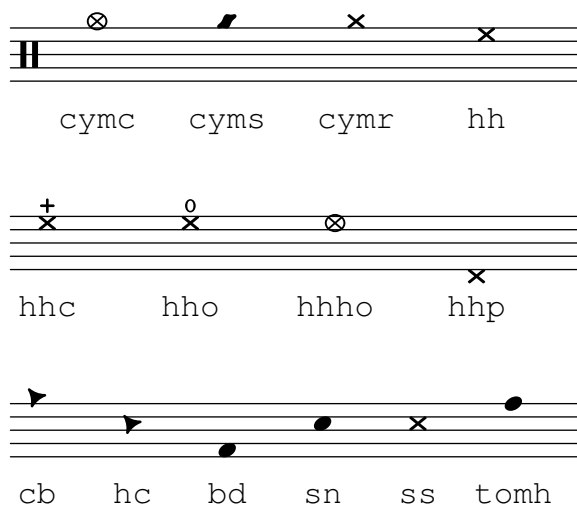
```

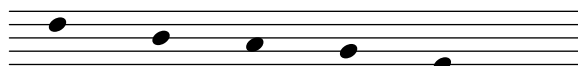


There are also other layout possibilities. To use these, set the property `drumStyleTable` in context `DrumVoice`. The following variables have been predefined:

`drums-style`

This is the default. It typesets a typical drum kit on a five-line staff:



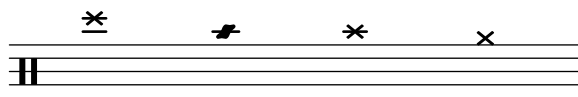


tommh tomml tom1 tomfh tomfl

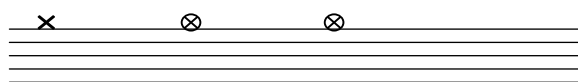
The drum scheme supports six different toms. When there are fewer toms, simply select the toms that produce the desired result. For example, to get toms on the three middle lines you use `tommh`, `tomml`, and `tomfh`.

agostini-drums-style

Invented by the French percussionist Dante Agostini in 1965, this notation is commonly employed in France but also elsewhere.



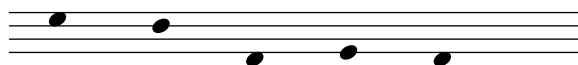
cymc cymh cymr hh



hhc hho hhho hhp



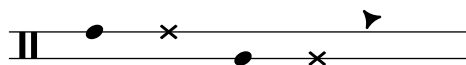
cb hc bd sn ss tomh



tommh tomml tom1 tomfh tomfl

timbales-style

This typesets timbales on a two line staff:



timh ssh timl ssl cb

congas-style

This typesets congas on a two line staff:



cgh cgho cghm ssh cgl cglo cglm ssl

bongos-style

This typesets bongos on a two line staff:



boh boho boh ssh bol bolo bolm ssl

percussion-style

To typeset all kinds of simple percussion on one line staves:



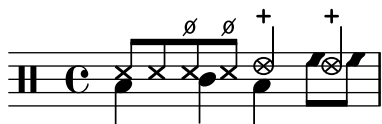
Custom percussion staves

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 }
>>
```



Fragments de codi seleccionats

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], pàgina 194.

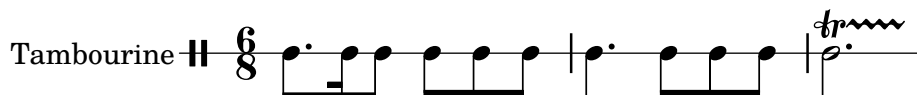
A tambourine, entered with ‘tamb’:

```
#(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"
}
```

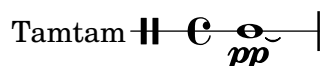
```

}

\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



Vegeu també

Snippets: Secció “Percussion” in *Fragments de codi*.

Internals Reference: Secció “DrumStaff” in *Referència de funcionament intern*, Secció “DrumVoice” in *Referència de funcionament intern*.

Ghost notes

Ghost notes for drums and percussion may be created using the `\parenthesize` command detailed in [Parentheses], pàgina 223.

```
\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
      }
    >>
  }
>>
```



Vegeu també

Snippets: Secció “Percussion” in *Fragments de codi*.

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 [Breath marks], pàgina 134.
- Legato playing is indicated by [Slurs], pàgina 130.
- Different types of tonguings, ranging from legato to non-legato to staccato are usually shown by articulation marks, sometimes combined with slurs, see [Articulations and ornamentations], pàgina 119, and Secció A.14 [List of articulations], pàgina 732.
- Flutter tonguing is usually indicated by placing a tremolo mark and a text markup on the note. See [Tremolo repeats], pàgina 162.

Other aspects of musical notation that can apply to wind instruments:

- Many wind instruments are transposing instruments, see <undefined> [Instrument transpositions], pàgina <undefined>.
- Slide glissandi are characteristic of the trombone, but other winds may perform keyed or valved glissandi. See [Glissando], pàgina 137.
- Harmonic series glissandi, which are possible on all brass instruments but common for French Horns, are usually written out as [Grace notes], pàgina 112.
- Pitch inflections at the end of a note are discussed in [Falls and doits], pàgina 136.
- Key slaps or valve slaps are often shown by the **cross** style of <undefined> [Special note heads], pàgina <undefined>.
- Woodwinds can overblow low notes to sound harmonics. These are shown by the **flageolet** articulation. See Secció A.14 [List of articulations], pàgina 732.
- 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 [Articulations and ornamentations], pàgina 119, and Secció A.14 [List of articulations], pàgina 732.
- Stopped horns are indicated by the **stopped** articulation. See [Articulations and ornamentations], pàgina 119.

Fragments de codi seleccionats

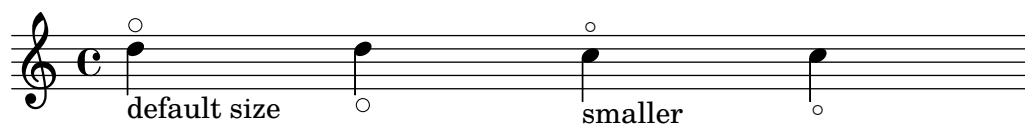
Changing \flageolet mark size

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
}
```



Vegeu també

Notation Reference: [Breath marks], pàgina 134, [Slurs], pàgina 130, [Articulations and ornamentations], pàgina 119, Secció A.14 [List of articulations], pàgina 732, [Tremolo repeats], pàgina 162, [\[undefined\]](#) [Instrument transpositions], pàgina [\[undefined\]](#), [Glissando], pàgina 137, [Grace notes], pàgina 112, [Falls and doits], pàgina 136, [\[undefined\]](#) [Special note heads], pàgina [\[undefined\]](#),

Snippets: Secció “Winds” in *Fragments de codi*.

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 Secció 2.6.3.1 [Woodwind diagrams], pàgina 396.

Fragments de codi seleccionats

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
  }
}
```



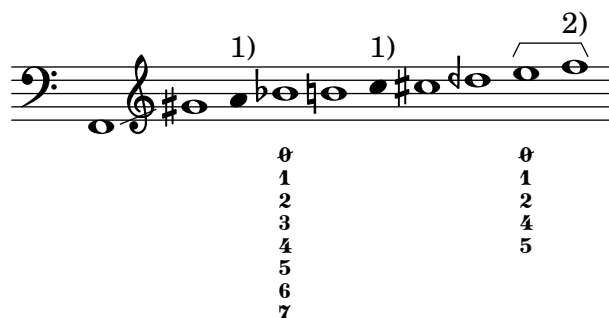
Recorder fingering chart

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
  }
}
```



Vegeu també

Notation Reference: Secció 2.6.3.1 [Woodwind diagrams], pàgina 396.

Snippets: Secció “Winds” in *Fragments de codi*.

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.

Vegeu també

Snippets: Secció “Winds” in *Fragments de codi*.

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.



Vegeu també

Snippets: Secció “Winds” in *Fragments de codi*.

2.6.3 Woodwinds

This section discusses notation specifically for woodwind instruments.

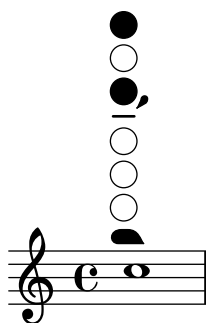
2.6.3.1 Woodwind diagrams

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 . ()))
  }
}

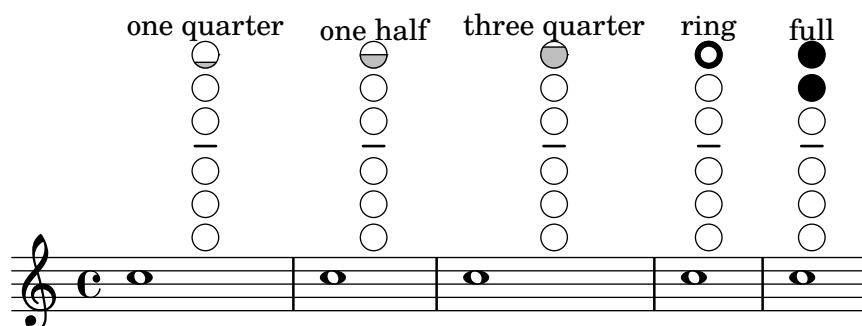
c''1^\markup {
  \center-column {
    "one half"
    \woodwind-diagram #'flute #'((cc . (one1h))
                                (lh . ()))
                                (rh . ()))
  }
}
```

}

```
c''1^\markup {
  \center-column {
    "three quarter"
    \woodwind-diagram #'flute #'((cc . (one3q))
                                (lh . ()))
                                (rh . ()))
  }
}
```

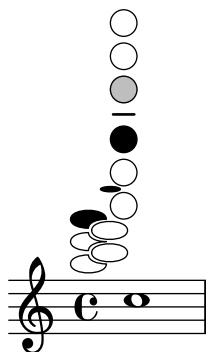
```
c''1^\markup {
  \center-column {
    "ring"
    \woodwind-diagram #'flute #'((cc . (oneR))
                                (lh . ()))
                                (rh . ()))
  }
}
```

```
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:

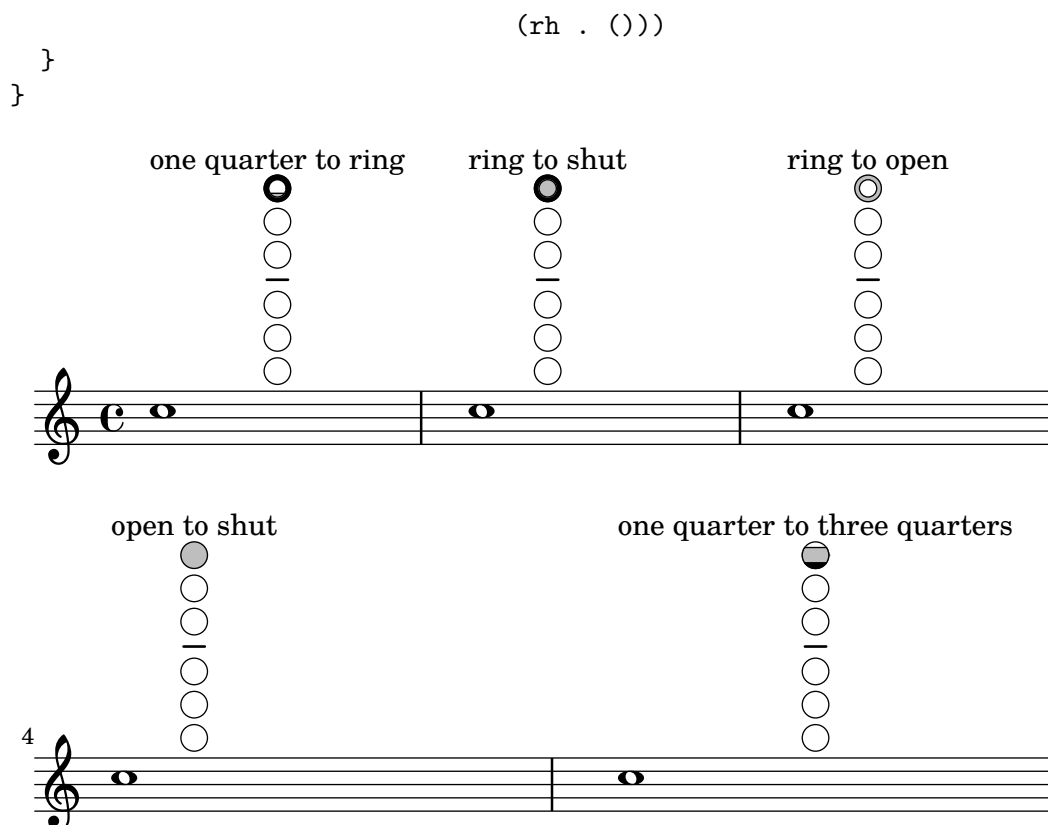
```
\textLengthOn
c''1^\markup {
  \center-column {
    "one quarter to ring"
    \woodwind-diagram #'flute #'((cc . (one1qTR))
                          (lh . ()))
                          (rh . ()))
  }
}

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 . ()))
  }
}
```

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`.

Instruccions predefinides

Fragments de codi seleccionats

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
    #'()
  }
}

```

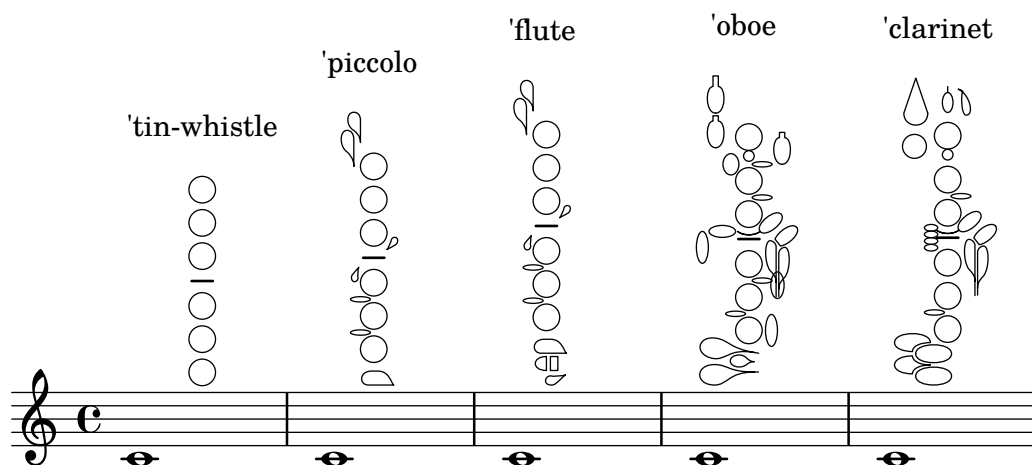
```

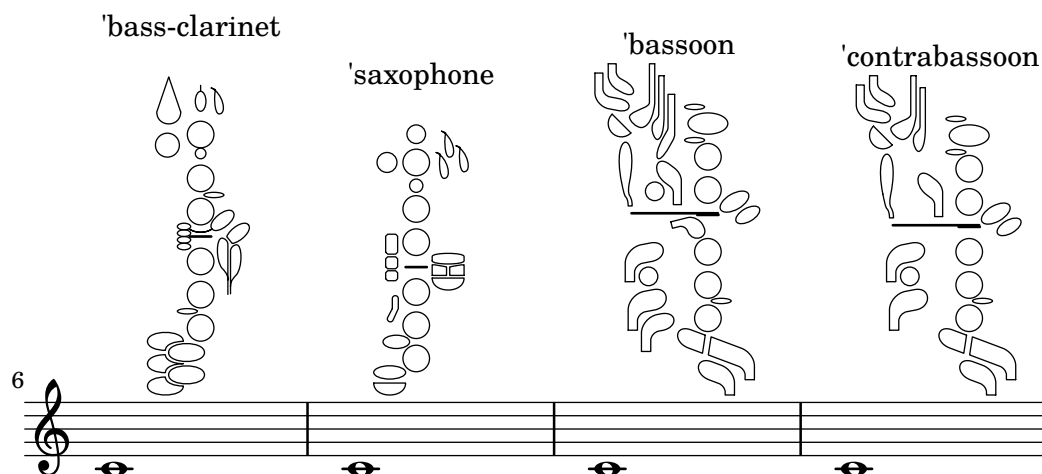
c1^\markup {
  \center-column {
    'saxophone
    " "
    \woodwind-diagram
    #'saxophone
    #'()
  }
}

c1^\markup {
  \center-column {
    'bassoon
    " "
    \woodwind-diagram
    #'bassoon
    #'()
  }
}

c1^\markup {
  \center-column {
    'contrabassoon
    " "
    \woodwind-diagram
    #'contrabassoon
    #'()
  }
}
}

```



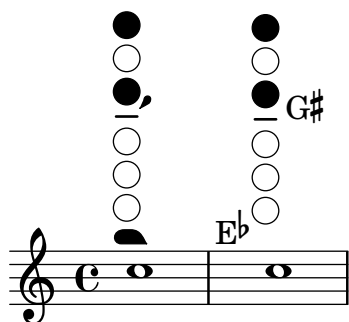


Graphical and text woodwind diagrams

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.

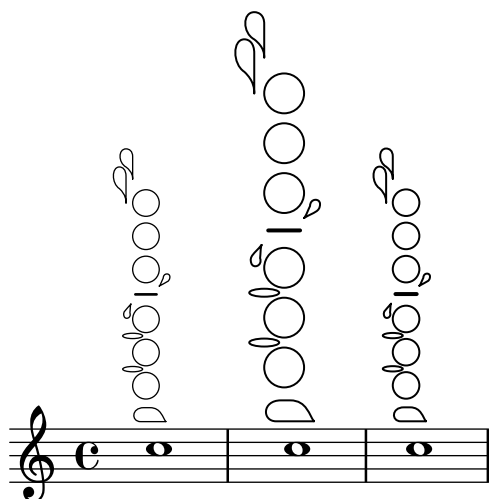
```
\relative c'' {
  \textLengthOn
  c1^\markup
    \woodwind-diagram
    #'piccolo
```

```

#'()

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}
```



Vegeu també

Installed Files: `scm/define-woodwind-diagrams.scm`,
`scm/display-woodwind-diagrams.scm`.

Snippets: Secció “Winds” in *Fragments de codi*.

Internals Reference: Secció “TextScript” in *Referència de funcionament intern*, Secció “instrument-specific-markup-interface” in *Referència de funcionament intern*.

2.7 Chord notation

1. Fair is the sun - shine, Fair - er the moon - light
 2. Fair are the mead - ows, Fair - er the wood - land,

And all the stars in heav'n a - bove;
 Robed in the flow - 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

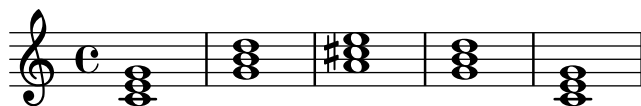
Chord mode is used to enter chords using an indicator of the chord structure, rather than the chord pitches.

Chord mode overview

Chords can be entered as simultaneous music, as discussed in [Chorded notes], pàgina 164.

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 Secció 5.4.1 [Input modes], pàgina 601.

```
\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 }
}
```



Vegeu també

Music Glossary: Secció “chord” in *Glossari musical*.

Notation Reference: [Chorded notes], pàgina 164, Secció 5.4.1 [Input modes], pàgina 601.

Snippets: Secció “Chords” in *Fragments de codi*.

Advertiments i problemes coneguts

Predefined shorthands for articulations and ornaments cannot be used on notes in chord mode, see [Articulations and ornamentations], pàgina 119.

Common chords

Major triads are entered by including the root and an optional duration:

```
\chordmode { c2 f4 g }
```



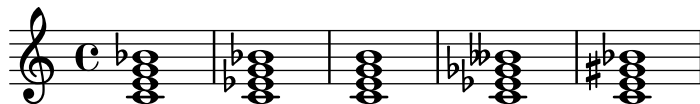
Minor, augmented, and diminished triads are entered by placing `:` and a quality modifier string after the duration:

```
\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 Secció A.2 [Common chord modifiers], pàgina 635.

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.	

Vegeu també

Notation Reference: Secció A.2 [Common chord modifiers], pàgina 635, [Extended and altered chords], pàgina 407.

Snippets: Secció “Chords” in *Fragments de codi*.

Advertiments i problemes coneguts

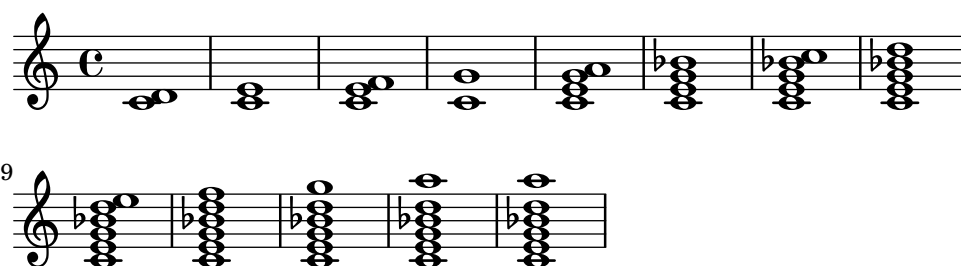
Only one quality modifier should be used per chord, typically on the highest step present in the chord. Chords with more than quality modifier will be parsed without an error or warning, but the results are unpredictable. Chords that cannot be achieved with a single quality modifier should be altered by individual pitches, as described in [Extended and altered chords], pàgina 407.

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.

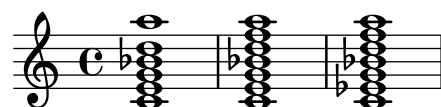
```
\chordmode {
  c1:2 c:3 c:4 c:5
  c1:6 c:7 c:8 c:9
  c1:10 c:11 c:12 c:13
  c1:14
}
```



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
}
```



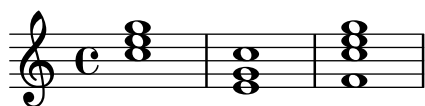
The modifier **sus** can be added to the modifier string to create suspended chords. This removes the 3rd step from the chord. Append either 2 or 4 to add the 2nd or 4th step to the chord. When **sus** is followed by either a 2nd or 4th step, it is equivalent to ^3, otherwise to **sus4**, namely 5.4.

```
\chordmode {
  c1:sus c:sus2 c:sus4 c:5.4
}
```



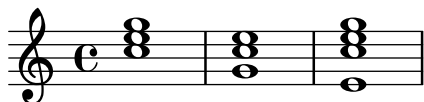
Inversions (putting a pitch other than the root on the bottom of the chord) and added bass notes can be specified by appending */pitch* to the chord.

```
\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 Secció A.2 [Common chord modifiers], pàgina 635.

Vegeu també

Notation Reference: Secció A.2 [Common chord modifiers], pàgina 635.

Snippets: Secció “Chords” in *Fragments de codi*.

Advertiments i problemes coneguts

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

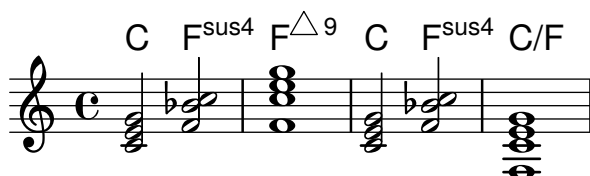
Chords can be entered as simultaneous notes or through the use of chord mode. The displayed chord name will be the same, regardless of the mode of entry, unless there are inversions or added bass notes:

```
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^Δ

```

\new ChordNames {
  \chordmode {
    c2 f4.:m g8:maj7
  }
}

```

C Fm G^Δ

Fragments de codi seleccionats

Showing chords at changes

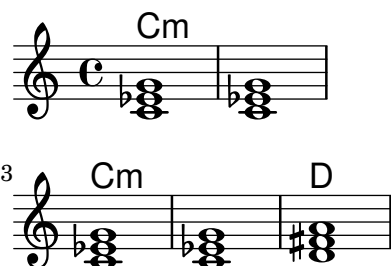
Chord names can be displayed only at the start of lines and when the chord changes.

```

harmonies = \chordmode {
  c1:m c:m \break c:m c:m d
}

<<
\new ChordNames {
  \set chordChanges = ##t
  \harmonies
}
\new Staff {
  \relative c' { \harmonies }
}
>>

```



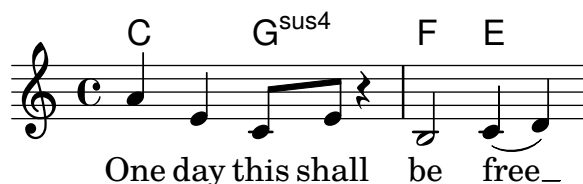
Simple lead sheet

When put together, chord names, a melody, and lyrics form a 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 __ }
>>

```



Vegeu també

Music Glossary: Secció “chord” in *Glossari musical*.

Notation Reference: [Writing music in parallel], pàgina 183.

Snippets: Secció “Chords” in *Fragments de codi*.

Internals Reference: Secció “ChordNames” in *Referència de funcionament intern*, Secció “ChordName” in *Referència de funcionament intern*, Secció “Chord_name_engraver” in *Referència de funcionament intern*, Secció “Volta_engraver” in *Referència de funcionament intern*, Secció “Bar_engraver” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

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 Secció “Literature list” in *Monografia*). 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 Secció A.1 [Chord name chart], pàgina 634.

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 [#] /B [#]	B ^b /B ^b
german	E/d	Cm	H/h	H [#] /his	B/b
semi-german	E/d	Cm	H/h	H [#] /his	B ^b /b
italian	Mi/Re	Do m	Si/Si	Si [#] /Si [#]	Si ^b /Si ^b
french	Mi/Ré	Do m	Si/Si	Si [#] /Si [#]	Si ^b /Si ^b



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
}
```

$C^{7\flat9}$ $C^{7\flat9}/G$

$C^{7/b9}$ $C^{7/b9}/G$

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 Fm⁷

C- F-⁷

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.

Instruccions predefinides

`\whiteTriangleMarkup`, `\blackTriangleMarkup`, `\germanChords`, `\semiGermanChords`, `\italianChords`, `\frenchChords`.

Fragments de codi seleccionats*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)" }
}

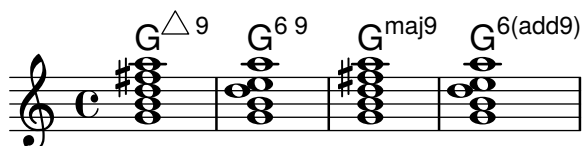
% Convert music to list and prepend to existing exceptions.
chExceptions = #( append
  ( sequential-music-to-chord-exceptions chExceptionMusic #t)
  ignatzekExceptions)

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+
}
```

$C^{\triangle} C^{j7}$

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
}
```

$F^{\triangle} \mid F^7 \mid B\flat^7 \mid$

Volta below chords

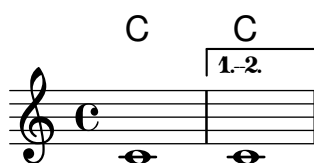
By adding the `Volta_engraver` to the relevant staff, volte can be put under chords.

```
\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
}

```

C⁷ sus4 **C**⁷ | sus4

Vegeu també

Notation Reference: Secció A.1 [Chord name chart], pàgina 634, Secció A.2 [Common chord modifiers], pàgina 635.

Essay on automated music engraving: Secció “Literature list” in *Monografia*.

Installed Files: `scm/chords-ignatzek.scm`, `scm/chord-entry.scm`,
`ly/chord-modifier-init.ly`.

Snippets: Secció “Chords” in *Fragments de codi*.

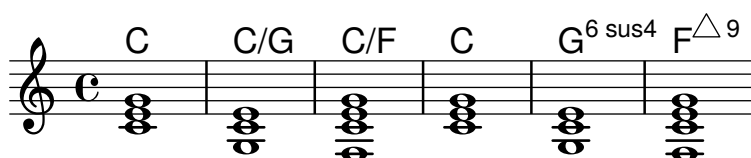
Advertiments i problemes coneguts

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 $\frac{6}{4}^+$ 6 #

5 $\frac{6}{4}$ 6 5 5 6 $\frac{6}{5}$ #

6 # 6 $\frac{6}{5}$ 5 $\frac{5}{4}$ 6 6 $\frac{6}{5}$ $\frac{5}{4}$ 3 5 # 7 6 5 $\frac{9}{4}$ $\frac{8}{3}$

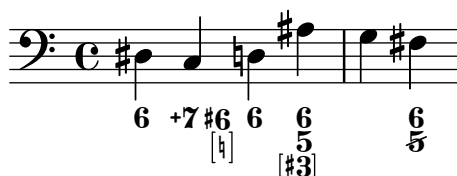
Figured bass notation can be displayed.

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.

Vegeu també

Music Glossary: Secció “figured bass” in *Glossari musical*.

Snippets: Secció “Chords” in *Fragments de codi*.

Entering figured bass

`\figuremode` is used to switch the input mode to figure mode. More information on different input modes can be found at Secció 5.4.1 [Input modes], pàgina 601.

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) may be used for modifying scale steps. These are entered by appending `+` (for sharps), `-` (for flats) or `!` (for naturals) after the number. For double accidentals the modifier is applied twice. For the modification of the third step the number is often omitted, which can be achieved by using `_` instead of a number.

```
\figures {
  <7! 6+ 4-> <5++> <3--> < _+ > < 7 _!>
}
```

7 **x5** **b3** **#** **7**
#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 6

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⁽¹⁾
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	$\flat 7 \times 5 \flat 3$ $\sharp 6$ $\flat 4$
\+, /	Augmented and diminished steps	$+6$ 7 5
\\	Raised sixth step	$\mathbf{6}$
\!	End of continuation line	



Instruccions predefinides

`\bassFigureExtendersOn`, `\bassFigureExtendersOff`.

Fragments de codi seleccionats

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 #5 6 **+6 5# 6** **6+ 5# 6** **6+ #5 6**
 $\flat 4$ $4 \flat$ $4 \flat$ $\flat 4$

Vegeu també

Snippets: Secció “Chords” in *Fragments de codi*.

Internals Reference: Secció “BassFigure” in *Referència de funcionament intern*, Secció “BassFigureAlignment” in *Referència de funcionament intern*, Secció “BassFigureLine” in *Referència de funcionament intern*, Secció “BassFigureBracket” in *Referència de funcionament intern*, Secció “BassFigureContinuation” in *Referència de funcionament intern*, Secció “FiguredBass” in *Referència de funcionament intern*.

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.

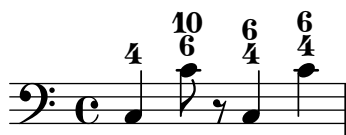
```
<<
  \relative {
    c' '4 c'8 r8 c,4 c'
  }
  \new FiguredBass {
    \figuremode {
      <4>4 <10 6>8 s8
      <6 4>4 <6 4>
    }
  }
>>
```



In the example above, the **FiguredBass** context must be explicitly instantiated to avoid creating a second (empty) staff.

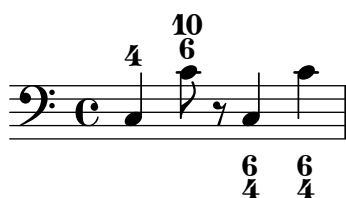
Figured bass can also be added to **Staff** contexts directly. In this case, the vertical position of the figures is adjusted automatically.

```
<<
  \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'
}
>>
```



Instruccions predefinides

`\bassFigureStaffAlignmentDown`,
`\bassFigureStaffAlignmentNeutral`.

`\bassFigureStaffAlignmentUp`,

Vegeu també

Snippets: Secció “Chords” in *Fragments de codi*.

Internals Reference: Secció “BassFigure” in *Referència de funcionament intern*, Secció “BassFigureAlignment” in *Referència de funcionament intern*, Secció “BassFigureLine” in *Referència de funcionament intern*, Secció “BassFigureBracket” in *Referència de funcionament intern*, Secció “BassFigureContinuation” in *Referència de funcionament intern*, Secció “FiguredBass” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

To ensure that continuation lines work properly, it is safest to use the same rhythm in the figure line as in the bass line.

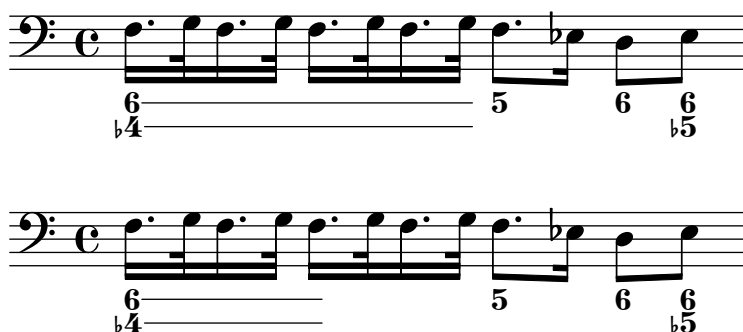
```
<<
{
  \clef bass
  \repeat unfold 4 { f16. g32 } f8. es16 d8 es
}
\figures {
  \bassFigureExtendersOn
  % The extenders are correct here, with the same rhythm as the bass
  \repeat unfold 4 { <6 4->16. <6 4->32 }
  <5>8. r16 <6>8 <6\! 5->
}
>>
```



```

>>
<<
{
  \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 [\[Note names in other languages\]](#), pàgina [\[undefined\]](#).
- Non-standard key signatures are addressed in [\[Key signature\]](#), pàgina [\[undefined\]](#).
- Contemporary practises in displaying accidentals are addressed in [\[Automatic accidentals\]](#), pàgina [\[undefined\]](#).

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], pàgina 65.
- Basic polymetric notation is addressed in [Polymetric notation], pàgina 75.
- Feathered beams are addressed in [Feathered beams], pàgina 96.
- Mensurstriche bar lines (bar lines between staves only) are addressed in [Grouping staves], pàgina 187.

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, fi- li- i He- vae. Ad te su- spi-

rá- mus, ge- mén- tes et flen- tes in hac la- cri-

The image shows three staves of ancient musical notation. Each staff consists of a single red line with square neumes placed above or below it. The notation is simple and lacks modern staff lines. The lyrics are in Latin and are aligned with the neumes. The first staff ends with a double bar line. The second staff also ends with a double bar line. The third staff ends with a double bar line.



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], pàgina 431,
- [Mensural accidentals and key signatures], pàgina 433,
- [Mensural rests], pàgina 432,
- [Mensural clefs], pàgina 429,
- [Gregorian clefs], pàgina 436,
- [Mensural flags], pàgina 432,
- [Mensural time signatures], pàgina 430.

Some notational concepts are introduced specifically for ancient notation,

- [Custodes], pàgina 427,
- [Divisiones], pàgina 437,
- [Ligatures], pàgina 427.

Vegeu també

Music Glossary: Secció “custos” in *Glossari musical*, Secció “ligature” in *Glossari musical*, Secció “mensural notation” in *Glossari musical*.

Notation Reference: [Mensural note heads], pàgina 431, [Mensural accidentals and key signatures], pàgina 433, [Mensural rests], pàgina 432, [Gregorian clefs], pàgina 436, [Mensural flags], pàgina 432, [Mensural time signatures], pàgina 430, [Custodes], pàgina 427, [Divisiones], pàgina 437, [Ligatures], pàgina 427.

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.

Vegeu també

Music Glossary: Secció “mensural notation” in *Glossari musical*, Secció “flag” in *Glossari musical*.

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], pàgina 435,
- [Mensural contexts], pàgina 428.

Vegeu també

Music Glossary: Secció “mensural notation” in *Glossari musical*.

Notation Reference: [Gregorian chant contexts], pàgina 435, [Mensural contexts], pàgina 428.

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], pàgina 434, and [Gregorian square neume ligatures], pàgina 439.

Vegeu també

Music Glossary: Secció “ligature” in *Glossari musical*.

Notation Reference: [White mensural ligatures], pàgina 434, [Gregorian square neume ligatures], pàgina 439.

Advertiments i problemes coneguts

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.

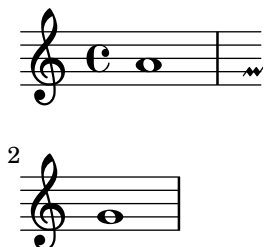
The syntax still uses the deprecated infix style `\[music expr \]`. For consistency reasons, it will eventually be changed to postfix style `note\[... note\]`.

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.

```
vaticana medicaea hufnagel mensural
```

Vegeu també

Music Glossary: Secció “custos” in *Glossari musical*.

Snippets: Secció “Ancient notation” in *Fragments de codi*.

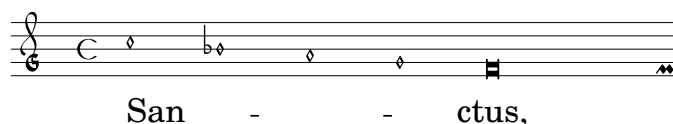
Internals Reference: Secció “Custos” in *Referència de funcionament intern*.

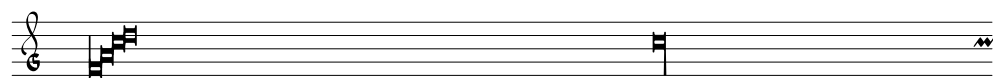
2.9.3 Typesetting mensural music

Mensural contexts

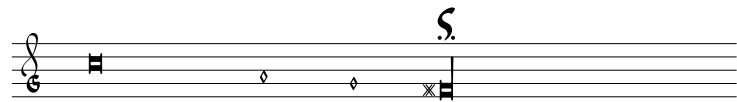
The predefined `MensuralVoice` and `MensuralStaff` contexts can be used to engrave a piece in mensural style. These contexts initialize all relevant context properties and grob properties to proper values, so you can immediately go ahead entering the chant, as the following excerpt demonstrates:

```
\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



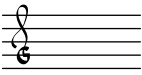
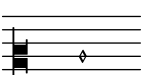
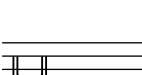
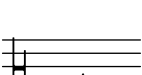
Vegeu també

Music Glossary: Secció “mensural notation” in *Glossari musical*.

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 [\[Clef\]](#), pàgina [\[Clef\]](#). 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>	

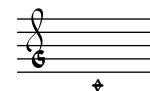
petrucci style F clefs, for use on different staff lines (the example shows the 3rd staff line F clef)

`petrucci-f3`, `petrucci-f4`,
`petrucci-f5`



petrucci style G clef

`petrucci-g`



Vegeu també

Music Glossary: Secció “mensural notation” in *Glossari musical*, Secció “clef” in *Glossari musical*.

Notation Reference: `\time` [Clef], pàgina `\time`.

Advertiments i problemes coneguts

The mensural g clef is mapped to the Petrucci g clef.

Mensural time signatures

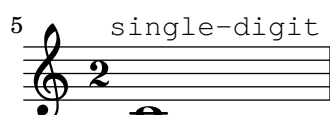
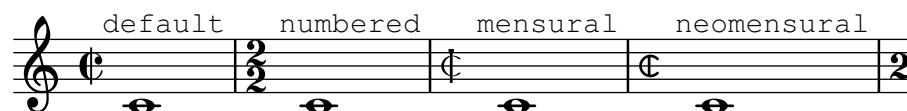
There is limited support for mensuration signs (which are similar to, but not exactly the same as time signatures). The glyphs are hard-wired to particular time fractions. In other words, to get a particular mensuration sign with the `\time n/m` command, `n` and `m` have to be chosen according to the following table

`\time 4/4` `\time 2/2` `\time 6/4` `\time 6/8`
C C C C

`\time 3/2` `\time 3/4` `\time 9/4` `\time 9/8`
O O O O

`\time 4/8` `\time 2/4`
C D

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], pàgina 65, gives a general introduction to the use of time signatures.

Vegeu també

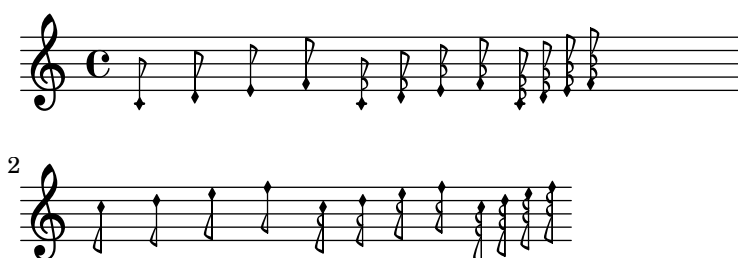
Music Glossary: Secció “mensural notation” in *Glossari musical*.

Notation Reference: [Time signature], pàgina 65.

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.

Vegeu també

Music Glossary: Secció “mensural notation” in *Glossari musical*, Secció “flag” in *Glossari musical*.

Advertiments i problemes coneguts

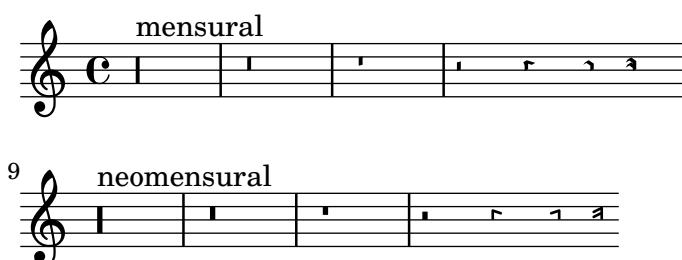
Vertically aligning each flag with a staff line assumes that stems always end either exactly on or exactly in the middle of two staff lines. This may not always be true when using advanced layout features of classical notation (which however are typically out of scope for mensural notation).

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.

Vegeu també

Music Glossary: Secció “mensural notation” in *Glossari musical*.

Notation Reference: [Rests], pàgina 57.

Snippets: Secció “Ancient notation” in *Fragments de codi*.

Advertiments i problemes coneguts

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
```

Vegeu també

Music Glossary: Secció “mensural notation” in *Glossari musical*, Secció “Pitch names” in *Glossari musical*, Secció “accidental” in *Glossari musical*, Secció “key signature” in *Glossari musical*.

Notation Reference: [\[Pitches\]](#), pàgina [\[Pitches\]](#), [\[Accidentals\]](#), pàgina [\[Accidentals\]](#), [\[Automatic accidentals\]](#), pàgina [\[Automatic accidentals\]](#), [\[Key signature\]](#), pàgina [\[Key signature\]](#).

Internals Reference: Secció “KeySignature” in *Referència de funcionament intern*.

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.

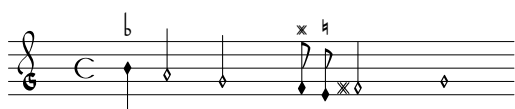
Support for such suggested accidentals is included, and can be switched on by setting `suggestAccidentals` to true.

```
\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
  }
}
```



Vegeu també

Internals Reference: Secció “Accidental_engraver” in *Referència de funcionament intern*, Secció “AccidentalSuggestion” in *Referència de funcionament intern*.

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`.

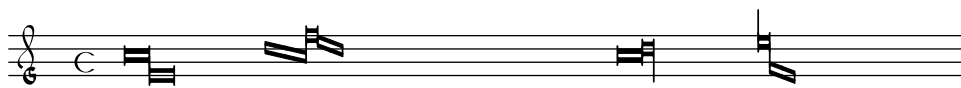
For example,

```
\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:



Vegeu també

Music Glossary: Secció “ligature” in *Glossari musical*.

Notation Reference: [Gregorian square neume ligatures], pàgina 439, [Ligatures], pàgina 427.

Advertiments i problemes coneguts

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.

Vegeu també

Music Glossary: Secció “ligature” in *Glossari musical*.

Notation Reference: [White mensural ligatures], pàgina 434, [Ligatures], pàgina 427.

Gregorian chant contexts

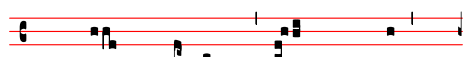
The predefined `VaticanaVoice` and `VaticanaStaff` can be used to engrave a piece of Gregorian chant in the style of the Editio Vaticana. These contexts initialize all relevant context properties and grob properties to proper values, so you can immediately go ahead entering the chant, as the following 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
    }
  >>
}

```




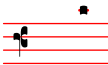


San- ctus, San- ctus,



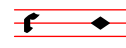
San- ctus

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 [\[Clef\]](#), pàgina [\[Clef\]](#). 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



Vegeu també

Music Glossary: Secció “clef” in *Glossari musical*.

Notation Reference: [\[Clef\]](#), pàgina [\[Clef\]](#).

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
```

Vegeu també

Music Glossary: Secció “accidental” in *Glossari musical*, Secció “key signature” in *Glossari musical*.

Notation Reference: [\[Pitches\]](#), pàgina [\[Pitches\]](#), [\[Accidentals\]](#), pàgina [\[Accidentals\]](#), [\[Automatic accidentals\]](#), pàgina [\[Automatic accidentals\]](#), [\[Key signature\]](#), pàgina [\[Key signature\]](#).

Internals Reference: Secció “KeySignature” in *Referència de funcionament intern*.

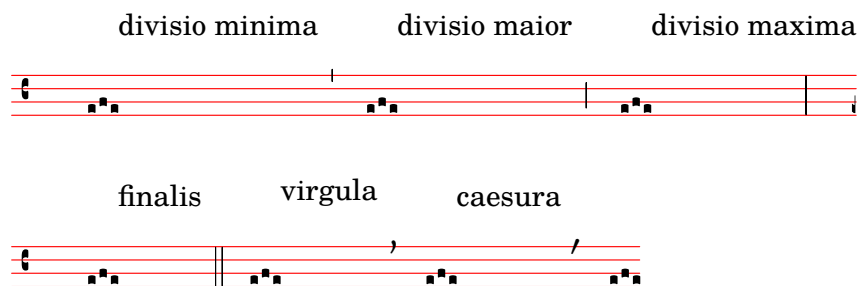
Divisiones

There are no rests in Gregorian chant notation; instead, it uses [\[Divisiones\]](#), pàgina [\[Divisiones\]](#).

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 [\[Breath marks\]](#), pàgina [\[Breath marks\]](#). 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`



Instruccions predefinides

`\virgula`, `\caesura`, `\divisioMinima`, `\divisioMaior`, `\divisioMaxima`, `\finalis`.

Vegeu també

Music Glossary: Secció “caesura” in *Glossari musical*, Secció “divisio” in *Glossari musical*.

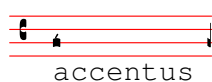
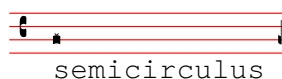
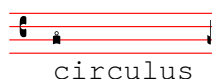
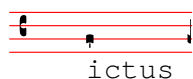
Notation Reference: [Breath marks], pàgina 134.

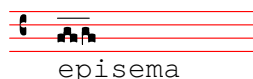
Installed Files: `ly/gregorian.ly`.

Gregorian articulation signs

In addition to the standard articulation signs described in section [Articulations and ornamentations], pàgina 119, 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
    \[ a_"episema" \epistemInitium \pes b \flexa a b \epistemFinis \flexa a \]
  }
}
```





episema

Vegeu també

Notation Reference: [Articulations and ornamentations], pàgina 119.

Snippets: Secció “Ancient notation” in *Fragments de codi*.

Internals Reference: Secció “Episema” in *Referència de funcionament intern*, Secció “EpisemaEvent” in *Referència de funcionament intern*, Secció “Episema_engraver” in *Referència de funcionament intern*, Secció “Script” in *Referència de funcionament intern*, Secció “ScriptEvent” in *Referència de funcionament intern*, Secció “Script_engraver” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

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
  }
}
```



Vegeu també

Notation Reference: [Breath marks], pàgina 134.

Internals Reference: Secció “BreathingSign” in *Referència de funcionament intern*.

Snippets: Secció “Ancient notation” in *Fragments de codi*.

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 curvation 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
Punctum		<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>
Punctum inclinatum		<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 $\backslash[a \backslash flexa g \backslash pes b \backslash]$ *Porrectus Auctus Descendens* $\backslash[a \backslash flexa g \backslash pes \backslash auctum \backslash descendens b \backslash]$ *Porrectus Deminutus* $\backslash[a \backslash flexa g \backslash pes \backslash deminutum b \backslash]$ **Climacus** $\backslash[\backslash virga b \backslash inclinatum a \backslash inclinatum g \backslash]$ *Climacus Auctus* $\backslash[\backslash virga b \backslash inclinatum a \backslash inclinatum \backslash auctum g \backslash]$ *Climacus Deminutus* $\backslash[\backslash virga b \backslash inclinatum a \backslash inclinatum \backslash deminutum g \backslash]$ **Scandicus** $\backslash[g \backslash pes a \backslash virga b \backslash]$ *Scandicus Auctus Descendens* $\backslash[g \backslash pes a \backslash pes \backslash auctum \backslash descendens b \backslash]$ *Scandicus Deminutus* $\backslash[g \backslash pes a \backslash pes \backslash deminutum b \backslash]$ **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)stroph**

\[\stroph a b \]

*Stroph Aucta*

\[\stroph \auctum b \]

**Bistroph**

\[\stroph a b \stroph a b \]



Tristropha


```
\[ \stropha b \stropha b
\stropha b \]
```

Trigonus


```
\[ \stropha b \stropha b
\stropha a \]
```

Instruccions predefinides

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.

Vegeu també

Music Glossary: Secció “ligature” in *Glossari musical*.

Notation Reference: [Gregorian square neume ligatures], pàgina 439, [White mensural ligatures], pàgina 434, [Ligatures], pàgina 427.

Advertiments i problemes coneguts

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" {
  -- -- -- -- .
}
>>
}

```



Vegeu també

Music Glossary: Secció “kievan notation” in *Glossari musical*.

Advertiments i problemes coneguts

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'

```



Vegeu també

Music Glossary: Secció “kievan notation” in *Glossari musical*, Secció “clef” in *Glossari musical*.

Notation Reference: [\[Clef\]](#), pàgina [\[Clef\]](#).

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
```



Vegeu també

Music Glossary: Secció “kievan notation” in *Glossari musical*, Secció “note head” in *Glossari musical*.

Notation Reference: Secció A.9 [Note head styles], pàgina 672.

Advertiments i problemes coneguts

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'
```



Vegeu també

Music Glossary: Secció “kievan notation” in *Glossari musical*, Secció “accidental” in *Glossari musical*.

Notation Reference: <undefined> [Accidentals], pàgina <undefined>, <undefined> [Automatic accidentals], pàgina <undefined>, Secció A.8 [The Feta font], pàgina 651,

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"
```



Vegeu també

Secció 1.2.5 [Bars], pàgina 97, Secció A.8 [The Feta font], pàgina 651,

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" {
      -- -- --
    }
  >>
}
```



Vegeu també

Music Glossary: Secció “ligature” in *Glossari musical*.

Notation Reference: [White mensural ligatures], pàgina 434, [Gregorian square neume ligatures], pàgina 439, [Ligatures], pàgina 427.

Advertiments i problemes coneguts

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
  }
}
```



Advertiments i problemes coneguts

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 = #""`.

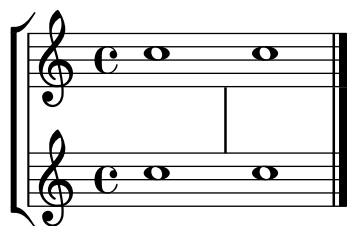
Mensurstriche layout

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.

The mensurstriche-layout where the bar lines do not show on the staves but between staves can be achieved with a **StaffGroup** instead of a **ChoirStaff**. The bar line on staves is blanked out using `\hide`.

```
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`.

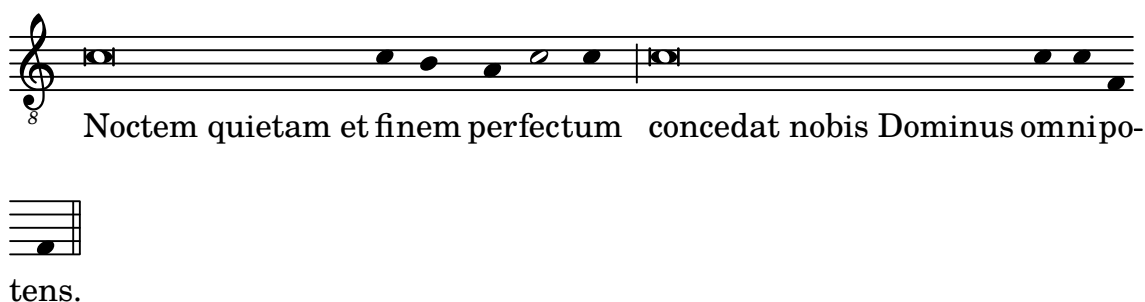
To remove the bar lines, the radical approach is to `\remove` the **Bar_engraver** from the **Staff** context. Again, one may want to use `\hide BarLine` instead, if an occasional barline is wanted.

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
    }
  }
}

```



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 [\[Writing pitches\]](#), [pàgina \[undefined\]](#), and the various note names that may be used are explained in [\[Note names in other languages\]](#), [pàgina \[undefined\]](#).

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 [Secció 2.10.2 \[Arabic music\]](#), [pàgina 457](#).

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 [Secció 2.10.3 \[Turkish classical music\]](#), [pàgina 462](#).

To locate init files such as `arabic.ly` or `makam.ly` on your system, see [Secció “Other sources of information”](#) in *Manual d’aprenentatge*.

Fragments de codi seleccionats

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
}
```



Vegeu també

Music Glossary: Secció “Common Practice Period” in *Glossari musical*, Secció “makamlar” in *Glossari musical*.

Learning Manual: Secció “Other sources of information” in *Manual d’aprenentatge*.

Notation Reference: <undefined> [Writing pitches], pàgina <undefined>, <undefined> [Note names in other languages], pàgina <undefined>, Secció 2.10.2 [Arabic music], pàgina 457, Secció 2.10.3 [Turkish classical music], pàgina 462.

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 Secció 2.10.1 [Common notation for non-Western music], pàgina 456.
- Additional key signatures can also be tailored as described in <undefined> [Key signature], pàgina <undefined>.

- Complex time signatures may require that notes be grouped manually as described in [Manual beams], pàgina 93.
- *Takasim* which are rhythmically free improvisations may be written down omitting bar lines as described in [Unmetered music], pàgina 73.

Vegeu també

Notation Reference: Secció 2.10.1 [Common notation for non-Western music], pàgina 456, [\[undefined\]](#) [Key signature], pàgina [\[undefined\]](#), [Manual beams], pàgina 93.

Snippets: Secció “World music” in *Fragments de codi*.

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], pàgina [\[undefined\]](#); the use of standard Western notation to notate non-Western music is discussed in Secció 2.10.1 [Common notation for non-Western music], pàgina 456.

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 `\dwn` 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 \dwn dob dobsb dodsd do do
}
```



Vegeu també

Notation Reference: [\[undefined\]](#) [Note names in other languages], pàgina [\[undefined\]](#), Secció 2.10.1 [Common notation for non-Western music], pàgina 456.

Snippets: Secció “World music” in *Fragments de codi*.

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

Fragments de codi seleccionats

Non-traditional key signatures

The commonly used `\key` command sets the `keyAlterations` property, in the `Staff` context.

To create non-standard key signatures, set this property directly. The format of this command is a list:

```
\set Staff.keyAlterations = #`(((octave . step) . alter) ((octave . step) .
alter) ...) where, for each element in the list, octave specifies the octave (0 being the
octave from middle C to the B above), step specifies the note within the octave (0 means C
and 6 means B), and alter is ,SHARP ,FLAT ,DOUBLE-SHARP etc. (Note the leading comma.)
```

Alternatively, for each item in the list, using the more concise format (`step . alter`) specifies that the same alteration should hold in all octaves.

For microtonal scales where a “sharp” is not 100 cents, `alter` refers to the alteration as a proportion of a 200-cent whole tone.

Here is an example of a possible key signature for generating a whole-tone scale:

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

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



Vegeu també

Music Glossary: Secció “maqam” in *Glossari musical*, Secció “bayati” in *Glossari musical*, Secció “rast” in *Glossari musical*, Secció “sukah” in *Glossari musical*, Secció “iraq” in *Glossari musical*, Secció “kurd” in *Glossari musical*.

Notation Reference: `<undefined>` [Key signature], pàgina `<undefined>`.

Learning Manual: Secció “Pitches and key signatures” in *Manual d’aprenentatge*.

Internals Reference: Secció “KeySignature” in *Referència de funcionament intern*.

Snippets: Secció “World music” in *Fragments de codi*, Secció “Pitches” in *Fragments de codi*.

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.

Fragments de codi seleccionats

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
}
```



Vegeu també

Music Glossary: Secció “semai” in *Glossari musical*, Secció “taqasim” in *Glossari musical*.

Notation Reference: [Manual beams], pàgina 93, [Automatic beams], pàgina 82, [Unmetered music], pàgina 73, [Automatic accidentals], pàgina 84, [Setting automatic beam behavior], pàgina 84, [Time signature], pàgina 65.

Snippets: Secció “World music” in *Fragments de codi*.

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"
  }
}
```



Vegeu també

Snippets: Secció “World music” in *Fragments de codi*.

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 Secció 2.10.1 [Common notation for non-Western music], pàgina 456.

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 Secció 2.10.1 [Common notation for non-Western music], pàgina 456.

Vegeu també

Music Glossary: Secció “makam” in *Glossari musical*, Secció “makamlar” in *Glossari musical*.

Notation Reference: Secció 2.10.1 [Common notation for non-Western music], pàgina 456.

3 General input and output

This section deals with general LilyPond input and output issues, rather than specific notation.

3.1 Input structure

The main format of input for LilyPond are text files. By convention, these files end with `.ly`.

3.1.1 Structure of a score

A `\score` block must contain a single music expression delimited by curly brackets:

```
\score {
  ...
}
```

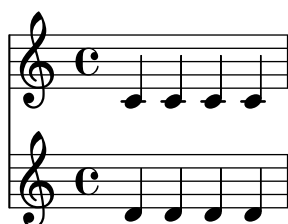
Nota: There must be **only one** outer music expression in a `\score` block, and it **must** be surrounded by curly brackets.

This single music expression may be of any size, and may contain other music expressions to any complexity. All of these examples are music expressions:

```
{ 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 { \flute }
      \new Staff { \oboe }
    >>
    \new StaffGroup <<
      \new Staff { \violinI }
      \new Staff { \violinII }
    >>
  >>
}
```

```
}
```

Comments are one exception to this general rule. (For others see Secció 3.1.5 [File structure], pàgina 468.) Both single-line comments and comments delimited by `%{ ... %}` may be placed anywhere within an input file. They may be placed inside or outside a `\score` block, and inside or outside the single music expression within a `\score` block.

Remember that even in a file containing only a `\score` block, it is implicitly enclosed in a `\book` block. A `\book` block in a source file produces at least one output file, and by default the name of the output file produced is derived from the name of the input file, so `fandangoforelephants.ly` will produce `fandangoforelephants.pdf`.

(For more details about `\book` blocks, see Secció 3.1.2 [Multiple scores in a book], pàgina 465, Secció 3.1.3 [Multiple output files from one input file], pàgina 466, Secció 3.1.5 [File structure], pàgina 468.)

Vegeu també

Learning Manual: Secció “Working on input files” in *Manual d’aprenentatge*, Secció “Music expressions explained” in *Manual d’aprenentatge*, Secció “Score is a (single) compound musical expression” in *Manual d’aprenentatge*.

3.1.2 Multiple scores in a book

A document may contain multiple pieces of music and text. Examples of these are an etude book, or an orchestral part with multiple movements. Each movement is entered with a `\score` block,

```
\score {
  ...music...
}
```

and texts are entered with a `\markup` block,

```
\markup {
  ...text...
}
```

All the movements and texts which appear in the same `.ly` file will normally be typeset in the form of a single output file.

```
\score {
  ...
}
\markup {
  ...
}
\score {
  ...
}
```

One important exception is within lilypond-book documents, where you explicitly have to add a `\book` block, otherwise only the first `\score` or `\markup` will appear in the output.

The header for each piece of music can be put inside the `\score` block. The `piece` name from the header will be printed before each movement. The title for the entire book can be put inside the `\book`, but if it is not present, the `\header` which is at the top of the file is inserted.

```
\header {
  title = "Eight miniatures"
  composer = "Igor Stravinsky"
}
```

```

\score {
  ...
  \header { piece = "Romanze" }
}
\markup {
  ...text of second verse...
}
\markup {
  ...text of third verse...
}
\score {
  ...
  \header { piece = "Menuetto" }
}

```

Pieces of music may be grouped into book parts using `\bookpart` blocks. Book parts are separated by a page break, and can start with a title, like the book itself, by specifying a `\header` block.

```

\bookpart {
  \header {
    title = "Book title"
    subtitle = "First part"
  }
  \score { ... }
  ...
}
\bookpart {
  \header {
    subtitle = "Second part"
  }
  \score { ... }
  ...
}

```

3.1.3 Multiple output files from one input file

If you want multiple output files from the same `.ly` file, then you can add multiple `\book` blocks, where each such `\book` block will result in a separate output file. If you do not specify any `\book` block in the input file, LilyPond will implicitly treat the whole file as a single `\book` block, see Secció 3.1.5 [File structure], pàgina 468.

When producing multiple files from a single source file, LilyPond ensures that none of the output files from any `\book` block overwrites the output file produced by a preceding `\book` from the same input file.

It does this by adding a suffix to the output name for each `\book` which uses the default output file name derived from the input source file.

The default behaviour is to append a version-number suffix for each name which may clash, so

```

\book {
  \score { ... }
  \paper { ... }
}
\book {

```

```

\score { ... }
\paper { ... }
}
\book {
  \score { ... }
  \paper { ... }
}

```

in source file `eightminiatures.ly` will produce

- `eightminiatures.pdf`,
- `eightminiatures-1.pdf` and
- `eightminiatures-2.pdf`.

3.1.4 Output file names

LilyPond provides facilities to allow you to control what file names are used by the various back-ends when producing output files.

In the previous section, we saw how LilyPond prevents name-clashes when producing several outputs from a single source file. You also have the ability to specify your own suffixes for each `\book` block, so for example you can produce files called `eightminiatures-Romanze.pdf`, `eightminiatures-Menuetto.pdf` and `eightminiatures-Nocturne.pdf` by adding a `\bookOutputSuffix` declaration inside each `\book` block.

```

\book {
  \bookOutputSuffix "Romanze"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputSuffix "Menuetto"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputSuffix "Nocturne"
  \score { ... }
  \paper { ... }
}

```

You can also specify a different output filename for `book` block, by using `\bookOutputName` declarations

```

\book {
  \bookOutputName "Romanze"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputName "Menuetto"
  \score { ... }
  \paper { ... }
}
\book {
  \bookOutputName "Nocturne"
  \score { ... }
}

```

```
\paper { ... }
}
```

The file above will produce these output files:

- Romanze.pdf,
- Menuetto.pdf and
- Nocturne.pdf.

3.1.5 File structure

A .ly file may contain any number of toplevel expressions, where a toplevel expression is one of the following:

- An output definition, such as `\paper`, `\midi`, and `\layout`. Such a definition at the toplevel changes the default book-wide settings. If more than one such definition of the same type is entered at the top level the definitions are combined, but in conflicting situations the later definitions take precedence. For details of how this affects the `\layout` block see Secció 4.2.1 [The `\layout` block], pàgina 531.
- A direct scheme expression, such as `#(set-default-paper-size "a7" 'landscape)` or `#(ly:set-option 'point-and-click #f)` .
- A `\header` block. This sets the global (i.e., the top of file) header block. This is the block containing the default settings of titling fields like composer, title, etc., for all books within the file (see [Titles explained], pàgina 470).
- A `\score` block. This score will be collected with other toplevel scores, and combined as a single `\book`. This behavior can be changed by setting the variable `toplevel-score-handler` at toplevel. The default handler is defined in the init file `../scm/lily.scm`.
- A `\book` block logically combines multiple movements (i.e., multiple `\score` blocks) in one document. If there are a number of `\scores`, one output file will be created for each `\book` block, in which all corresponding movements are concatenated. The only reason to explicitly specify `\book` blocks in a .ly file is if you wish to create multiple output files from a single input file. One exception is within lilypond-book documents, where you explicitly have to add a `\book` block if you want more than a single `\score` or `\markup` in the same example. This behavior can be changed by setting the variable `toplevel-book-handler` at toplevel. The default handler is defined in the init file `../scm/lily.scm`.
- A `\bookpart` block. A book may be divided into several parts, using `\bookpart` blocks, in order to ease the page breaking, or to use different `\paper` settings in different parts.
- A compound music expression, such as

```
{ c'4 d' e'2 }
```

This will add the piece in a `\score` and format it in a single book together with all other toplevel `\scores` and music expressions. In other words, a file containing only the above music expression will be translated into

```
\book {
  \score {
    \new Staff {
      \new Voice {
        { c'4 d' e'2 }
      }
    }
    \layout { }
  }
  \paper { }
```

```
\header { }
}
```

This behavior can be changed by setting the variable `toplevel-music-handler` at `toplevel`. The default handler is defined in the init file `../scm/lily.scm`.

- A markup text, a verse for example

```
\markup {
  2. The first line verse two.
}
```

Markup texts are rendered above, between or below the scores or music expressions, wherever they appear.

- A variable, such as

```
foo = { c4 d e d }
```

This can be used later on in the file by entering `\foo`. The name of a variable should have alphabetic characters only; no numbers, underscores or dashes.

The following example shows three things that may be entered at `toplevel`

```
\layout {
  % Don't justify the output
  ragged-right = ##t
}
```

```
\header {
  title = "Do-re-mi"
}
```

```
{ c'4 d' e2 }
```

At any point in a file, any of the following lexical instructions can be entered:

- `\version`
- `\include`
- `\sourcefilename`
- `\sourcefileline`
- A single-line comment, introduced by a leading `%` sign.
- A multi-line comment delimited by `%{ ... %}`.

Whitespace between items in the input stream is generally ignored, and may be freely omitted or extended to enhance readability. However, whitespace should always be used in the following circumstances to avoid errors:

- Around every opening and closing curly bracket.
- After every command or variable, i.e., every item that begins with a `\` sign.
- After every item that is to be interpreted as a Scheme expression, i.e., every item that begins with a `#` sign.
- To separate all elements of a Scheme expression.
- In `lyricmode` before and after `\set` and `\override` commands.

Vegeu també

Learning Manual: Secció “How LilyPond input files work” in *Manual d’aprenentatge*.

Notation Reference: [Titles explained], pàgina 470, Secció 4.2.1 [The `\layout` block], pàgina 531.

3.2 Titles and headers

Almost all printed music includes a title and the composer's name; some pieces include a lot more information.

3.2.1 Creating titles headers and footers

Titles explained

Each `\book` block in a single input file produces a separate output file, see Secció 3.1.5 [File structure], pàgina 468. Within each output file three types of titling areas are provided: *Book Titles* at the beginning of each book, *Bookpart Titles* at the beginning of each bookpart and *Score Titles* at the beginning of each score.

Values of titling fields such as `title` and `composer` are set in `\header` blocks. (For the syntax of `\header` blocks and a complete list of the fields available by default see [Default layout of bookpart and score titles], pàgina 473). Book Titles, Bookpart Titles and Score Titles can all contain the same fields, although by default the fields in Score Titles are limited to `piece` and `opus`.

`\header` blocks may be placed in four different places to form a descending hierarchy of `\header` blocks:

- At the top of the input file, before all `\book`, `\bookpart`, and `\score` blocks.
- Within a `\book` block but outside all the `\bookpart` and `\score` blocks within that book.
- Within a `\bookpart` block but outside all `\score` blocks within that bookpart.
- After the music expression in a `\score` block.

The values of the fields filter down this hierarchy, with the values set higher in the hierarchy persisting unless they are over-ridden by a value set lower in the hierarchy, so:

- A Book Title is derived from fields set at the top of the input file, modified by fields set in the `\book` block. The resulting fields are used to print the Book Title for that book, providing that there is other material which generates a page at the start of the book, before the first bookpart. A single `\pageBreak` will suffice.
- A Bookpart Title is derived from fields set at the top of the input file, modified by fields set in the `\book` block, and further modified by fields set in the `\bookpart` block. The resulting values are used to print the Bookpart Title for that bookpart.
- A Score Title is derived from fields set at the top of the input file, modified by fields set in the `\book` block, further modified by fields set in the `\bookpart` block and finally modified by fields set in the `\score` block. The resulting values are used to print the Score Title for that score. Note, though, that only `piece` and `opus` fields are printed by default in Score Titles unless the `\paper` variable, `print-all-headers`, is set to `#t`.

Nota: Remember when placing a `\header` block inside a `\score` block, that the music expression must come before the `\header` block.

It is not necessary to provide `\header` blocks in all four places: any or even all of them may be omitted. Similarly, simple input files may omit the `\book` and `\bookpart` blocks, leaving them to be created implicitly.

If the book has only a single score, the `\header` block should normally be placed at the top of the file so that just a Bookpart Title is produced, making all the titling fields available for use.

If the book has multiple scores a number of different arrangements of `\header` blocks are possible, corresponding to the various types of musical publications. For example, if the publication contains several pieces by the same composer a `\header` block placed at the top of the file

specifying the book title and the composer with `\header` blocks in each `\score` block specifying the piece and/or opus would be most suitable, as here:

```
\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.



More complicated arrangements are possible. For example, text fields from the `\header` block in a book can be displayed in all Score Titles, with some fields over-ridden and some manually suppressed:

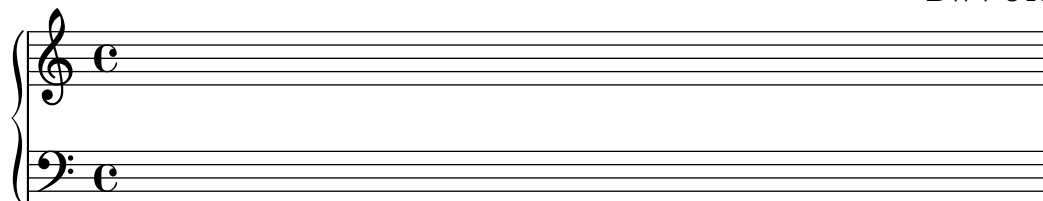
```
\book {
  \paper {
    print-all-headers = ##t
  }
  \header {
    title = "DAS WOHLTEMPERIRTE CLAVIER"
    subtitle = "TEIL I"
    % Do not display the default LilyPond footer for this book
    tagline = ##f
  }
  \markup { \vspace #1 }
  \score {
    \new PianoStaff <<
      \new Staff { s1 }
      \new Staff { \clef "bass" s1 }
    >>
    \header {
      title = "PRAELUDIUM I"
      opus = "BWV 846"
      % Do not display the subtitle for this score
      subtitle = ##f
    }
  }
  \score {
    \new PianoStaff <<
      \new Staff { s1 }
      \new Staff { \clef "bass" s1 }
    >>
    \header {
      title = "FUGA I"
      subsubtitle = "A 4 VOICI"
      opus = "BWV 846"
      % Do not display the subtitle for this score
      subtitle = ##f
    }
  }
}
```

DAS WOHLTEMPERIRTE CLAVIER

TEIL I

PRAELUDIUM I

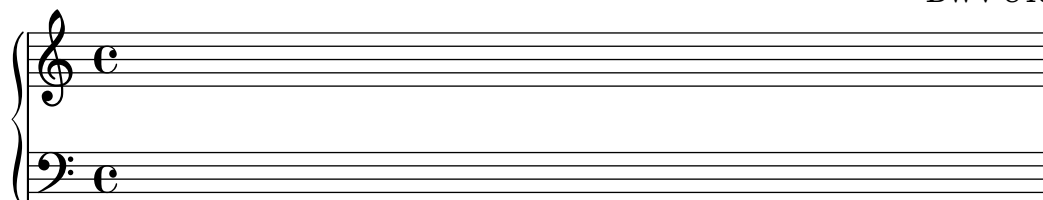
BWV 846



FUGA I

A 4 VOCI

BWV 846



Vegeu també

Notation Reference: Secció 3.1.5 [File structure], pàgina 468, [Default layout of bookpart and score titles], pàgina 473, [Custom layout for titles], pàgina 478.

Default layout of bookpart and score titles

This example demonstrates all printed `\header` variables:

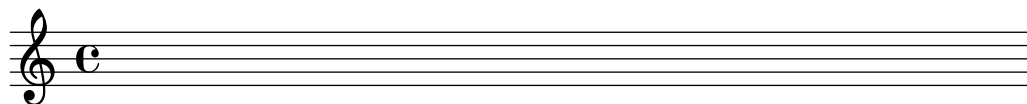
```
\book {
  \header {
    % The following fields are centered
    dedication = "Dedication"
    title = "Title"
    subtitle = "Subtitle"
    subsubtitle = "Subsubtitle"
    % The following fields are evenly spread on one line
    % the field "instrument" also appears on following pages
    instrument = \markup \with-color #green "Instrument"
    poet = "Poet"
    composer = "Composer"
    % The following fields are placed at opposite ends of the same line
    meter = "Meter"
    arranger = "Arranger"
    % The following fields are centered at the bottom
    tagline = "The tagline goes at the bottom of the last page"
    copyright = "The copyright goes at the bottom of the first page"
  }
  \score {
```

```

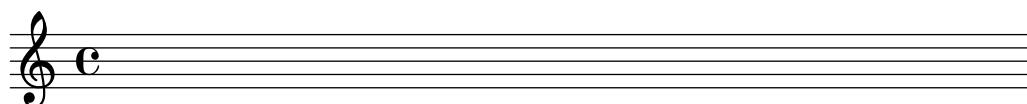
{ s1 }
\header {
  % The following fields are placed at opposite ends of the same line
  piece = "Piece 1"
  opus = "Opus 1"
}
}
\score {
  { s1 }
  \header {
    % The following fields are placed at opposite ends of the same line
    piece = "Piece 2 on the same page"
    opus = "Opus 2"
  }
}
\pageBreak
\score {
  { s1 }
  \header {
    % The following fields are placed at opposite ends of the same line
    piece = "Piece 3 on a new page"
    opus = "Opus 3"
  }
}
}

```

	Dedication	
	Title	
	Subtitle	
	Subsubtitle	
Poet	Instrument	Composer
Meter		Arranger
Piece 1		Opus 1

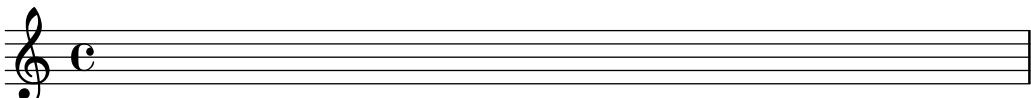


Piece 2 on the same page	Opus 2
--------------------------	--------



The copyright goes at the bottom of the first page

2	Instrument	
Piece 3 on a new page		Opus 3



The tagline goes at the bottom of the last page

Note that

- The instrument name will be repeated on every page.
- Only `piece` and `opus` are printed in a `\score` when the paper variable `print-all-headers` is set to `##f` (the default).
- Text fields left unset in a `\header` block are replaced with `\null` markups so that the space is not wasted.
- The default settings for `scoreTitleMarkup` place the `piece` and `opus` text fields at opposite ends of the same line.

To change the default layout see [Custom layout for titles], pàgina 478.

If a `\book` block starts immediately with a `\bookpart` block, no Book Title will be printed, as there is no page on which to print it. If a Book Title is required, begin the `\book` block with some markup material or a `\pageBreak` command.

Use the `breakbefore` variable inside a `\header` block that is itself in a `\score` block, to make the higher-level `\header` block titles appear on the first page on their own, with the music (defined in the `\score` block) starting on the next.

```
\book {
  \header {
    title = "This is my Title"
    subtitle = "This is my Subtitle"
    copyright = "This is the bottom of the first page"
  }
  \score {
    \repeat unfold 4 { e'' e'' e'' e'' }
    \header {
      piece = "This is the Music"
      breakbefore = ##t
    }
  }
}
```

This is my Title

This is my Subtitle

This is the bottom of the first page

2

This is the Music



Music engraving by LilyPond 2.19.48—www.lilypond.org

Vegeu també

Learning Manual: Secció “How LilyPond input files work” in *Manual d’aprenentatge*,

Notation Reference: [Custom layout for titles], pàgina 478, Secció 3.1.5 [File structure], pàgina 468.

Installed Files: `ly/titling-init.ly`.

Default layout of headers and footers

Headers and *footers* are lines of text appearing at the top and bottom of pages, separate from the main text of a book. They are controlled by the following `\paper` variables:

- `oddHeaderMarkup`
- `evenHeaderMarkup`
- `oddFooterMarkup`
- `evenFooterMarkup`

These markup variables can only access text fields from top-level `\header` blocks (which apply to all scores in the book) and are defined in `ly/titling-init.ly`. By default:

- page numbers are automatically placed on the top far left (if even) or top far right (if odd), starting from the second page.

- the `instrument` text field is placed in the center of every page, starting from the second page.
- the `copyright` text is centered on the bottom of the first page.
- the `tagline` is centered on the bottom of the last page, and below the `copyright` text if there is only a single page.

The default LilyPond footer text can be changed by adding a `tagline` in the top-level `\header` block.

```
\book {
  \header {
    tagline = "... music notation for Everyone"
  }
  \score {
    \relative {
      c'4 d e f
    }
  }
}
```



... music notation for Everyone

To remove the default LilyPond footer text, the `tagline` can be set to `##f`.

3.2.2 Custom titles headers and footers

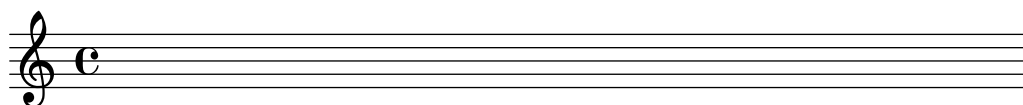
Custom text formatting for titles

Standard `\markup` commands can be used to customize any header, footer and title text within the `\header` block.

```
\score {
  { s1 }
  \header {
    piece = \markup { \fontsize #4 \bold "PRAELUDIUM I" }
    opus = \markup { \italic "BWV 846" }
  }
}
```

PRAELUDIUM I

BWV 846



Vegeu també

Notation Reference: Secció 1.8.2 [Formatting text], pàgina 237.

Custom layout for titles

`\markup` commands in the `\header` block are useful for simple text formatting, but they do not allow precise control over the placement of titles. To customize the placement of the text fields, change either or both of the following `\paper` variables:

- `bookTitleMarkup`
- `scoreTitleMarkup`

The placement of titles when using the default values of these `\markup` variables is shown in the examples in [Default layout of bookpart and score titles], pàgina 473.

The default settings for `scoreTitleMarkup` as defined in `ly/titling-init.ly` are:

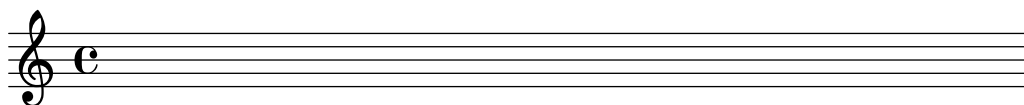
```
scoreTitleMarkup = \markup { \column {
  \on-the-fly \print-all-headers { \bookTitleMarkup \hspace #1 }
  \fill-line {
    \fromproperty #'header:piece
    \fromproperty #'header:opus
  }
}
```

This places the `piece` and `opus` text fields at opposite ends of the same line:

```
\score {
  { s1 }
  \header {
    piece = "PRAELUDIUM I"
    opus = "BWV 846"
  }
}
```

PRAELUDIUM I

BWV 846



This example redefines `scoreTitleMarkup` so that the `piece` text field is centered and in a large, bold font.

```
\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \fontsize #4 \bold \fromproperty #'header:piece
        \fromproperty #'header:opus
      }
    }
  }
}
```



```

\score {
  { s1 }
  \header {
    piece = "PRAELUDIUM I"
    opus = "BWV 846"
  }
}

```



Text fields not normally effective in score `\header` blocks can be printed in the Score Title area if `print-all-headers` is placed inside the `\paper` block. A disadvantage of using this method is that text fields that are intended specifically for the Bookpart Title area need to be manually suppressed in every `\score` block. See [Titles explained], pàgina 470.

To avoid this, add the desired text field to the `scoreTitleMarkup` definition. In the following example, the `composer` text field (normally associated with `bookTitleMarkup`) is added to `scoreTitleMarkup`, allowing each score to list a different composer:

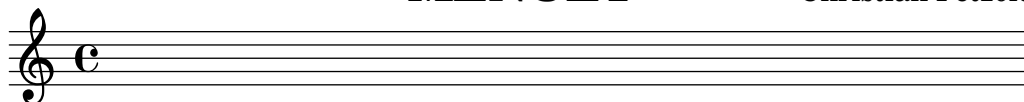
```

\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"
  }
}
}

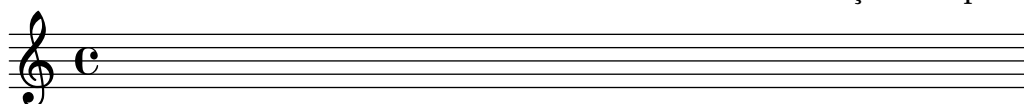
```

MENUET

Christian Petzold

**RONDEAU**

François Couperin



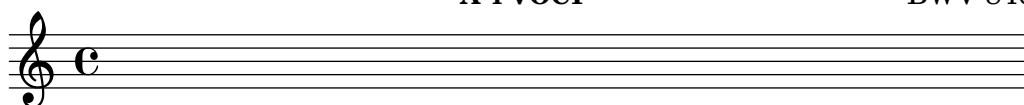
It is also possible to create your own custom text fields, and refer to them in the markup definition.

```
\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \override #`(direction . ,UP) {
          \dir-column {
            \center-align \fontsize #-1 \bold
            \fromproperty #'header:mycustomtext %% User-defined field
            \center-align \fontsize #4 \bold
            \fromproperty #'header:piece
          }
        }
        \fromproperty #'header:opus
      }
    }
  }
}
\header { tagline = ##f }
\score {
  { s1 }
  \header {
    piece = "FUGA I"
    mycustomtext = "A 4 VOICI" %% User-defined field
    opus = "BWV 846"
  }
}
}
```

FUGA I

A 4 VOICI

BWV 846



Vegeu també

Notation Reference: [Titles explained], pàgina 470.

Custom layout for headers and footers

`\markup` commands in the `\header` block are useful for simple text formatting, but they do not allow precise control over the placement of headers and footers. To customize the placement of the text fields, use either or both of the following `\paper` variables:

- `oddHeaderMarkup`
- `evenHeaderMarkup`
- `oddFooterMarkup`
- `evenFooterMarkup`

The `\markup` command `\on-the-fly` can be used to add markup conditionally to header and footer text defined within the `\paper` block, using the following syntax:

```
variable = \markup {
  ...
  \on-the-fly \procedure markup
  ...
}
```

The *procedure* is called each time the `\markup` command in which it appears is evaluated. The *procedure* should test for a particular condition and interpret (i.e., print) the *markup* argument if and only if the condition is true.

A number of ready-made procedures for testing various conditions are provided:

Procedure name	Condition tested
<code>print-page-number-check-first</code>	should this page number be printed?
<code>create-page-number-stencil</code>	<code>print-page-numbers true?</code>
<code>print-all-headers</code>	<code>print-all-headers true?</code>
<code>first-page</code>	first page in the book?
<code>(on-page nmbr)</code>	page number = nmbr?
<code>last-page</code>	last page in the book?
<code>not-first-page</code>	not first page in the book?
<code>part-first-page</code>	first page in the book part?
<code>part-last-page</code>	last page in the book part?
<code>not-single-page</code>	pages in book part > 1?

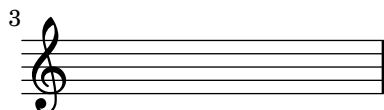
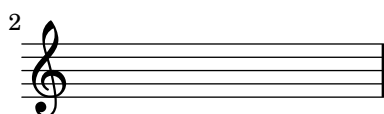
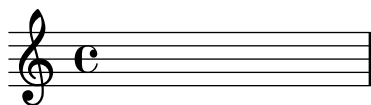
The following example centers page numbers at the bottom of every page. First, the default settings for `oddHeaderMarkup` and `evenHeaderMarkup` are removed by defining each as a *null* markup. Then, `oddFooterMarkup` is redefined with the page number centered. Finally, `evenFooterMarkup` is given the same layout by defining it as `\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

Several `\on-the-fly` conditions can be combined with an ‘and’ operation, for example,

```

\on-the-fly \first-page
\on-the-fly \last-page
{ \markup ... \fromproperty #'header: ... }

```

determines if the output is a single page.

Vegeu també

Notation Reference: [Titles explained], pàgina 470, [Default layout of bookpart and score titles], pàgina 473.

Installed Files: `../ly/titling-init.ly`.

3.2.3 Creating output file metadata

In addition to being shown in the printed output, `\header` variables are also used to set metadata for output files. For example, with PDF files, this metadata could be displayed by PDF readers as the **properties** of the PDF file. For each type of output file, only the `\header` definitions of blocks that define separate files of that type, and blocks higher in the block hierarchy, will be consulted. Therefore, for PDF files, only the `\book` level and the top level `\header` definitions affect the document-wide PDF metadata, whereas for MIDI files, all headers above or at the `\score` level are used.

For example, setting the `title` property of the `header` block to ‘Symphony I’ will also give this title to the PDF document, and use it as the sequence name of the MIDI file.

```

\header{
  title = "Symphony I"
}

```

If you want to set the title of the printed output to one value, but have the title property of the PDF to have a different value, you can use `pdftitle`, as below.

```
\header{
  title = "Symphony I"
  pdftitle = "Symphony I by Beethoven"
}
```

The variables `title`, `subject`, `keywords`, `subtitle`, `composer`, `arranger`, `poet`, `author` and `copyright` all set PDF properties and can all be prefixed with ‘pdf’ to set a PDF property to a value different from the printed output.

The PDF property `Creator` is automatically set to ‘LilyPond’ plus the current LilyPond version, and `CreationDate` and `ModDate` are both set to the current date and time. `ModDate` can be overridden by setting the header variable `moddate` (or `pdfmoddate`) to a valid PDF date string.

The `title` variable sets also the sequence name for MIDI. The `midititle` variable can be used to set the sequence name independently of the value used for typeset output.

3.2.4 Creating footnotes

Footnotes may be used in many different situations. In all cases, a ‘footnote mark’ is placed as a reference in text or music, and the corresponding ‘footnote text’ appears at the bottom of the same page.

Footnotes within music expressions and footnotes in stand-alone text outside music expressions are created in different ways.

Footnotes in music expressions

Music footnotes overview

Footnotes in music expressions fall into two categories:

Event-based footnotes

are attached to a particular event. Examples for such events are single notes, articulations (like fingering indications, accents, dynamics), and post-events (like slurs and manual beams). The general form for event-based footnotes is as follows:

```
[direction] \footnote [mark] offset footnote music
```

Time-based footnotes

are bound to a particular point of time in a musical context. Some commands like `\time` and `\clef` don’t actually use events for creating objects like time signatures and clefs. Neither does a chord create an event of its own: its stem or flag is created at the end of a time step (nominally through one of the note events inside). Exactly which of a chord’s multiple note events will be deemed the root cause of a stem or flag is undefined. So for annotating those, time-based footnotes are preferable as well.

A time-based footnote allows such layout objects to be annotated without referring to an event. The general form for Time-based footnotes is:

```
\footnote [mark] offset footnote [Context].GrobName
```

The elements for both forms are:

- | | |
|------------------|---|
| <i>direction</i> | If (and only if) the <code>\footnote</code> is being applied to a post-event or articulation, it must be preceded with a direction indicator (<code>-</code> , <code>_</code> , <code>^</code>) in order to attach <i>music</i> (with a footnote mark) to the preceding note or rest. |
| <i>mark</i> | is a markup or string specifying the footnote mark which is used for marking both the reference point and the footnote itself at the bottom of the page. It may be omitted (or equivalently replaced with <code>\default</code>) in which case a number in sequence will be |

generated automatically. Such numerical sequences restart on each page containing a footnote.

offset is a number pair such as ‘#(2 . 1)’ specifying the X and Y offsets in units of staff-spaces from the boundary of the object where the mark should be placed. Positive values of the offsets are taken from the right/top edge, negative values from the left/bottom edge and zero implies the mark is centered on the edge.

Context is the context in which the grob being footnoted is created. It may be omitted if the grob is in a bottom context, e.g., a **Voice** context.

GrobName specifies a type of grob to mark (like ‘Flag’). If it is specified, the footnote is not attached to a music expression in particular, but rather to all grobs of the type specified which occur at that moment of musical time.

footnote is the markup or string specifying the footnote text to use at the bottom of the page.

music is the music event or post-event or articulation that is being annotated.

Event-based footnotes

A footnote may be attached to a layout object directly caused by the event corresponding to *music* with the syntax:

```
\footnote [mark] offset footnote music
\book {
  \header { tagline = ##f }
  \relative c'' {
    \footnote #'(-1 . 3) "A note" a4
    a4
    \footnote #'(2 . 2) "A rest" r4
    a4
  }
}
```



¹A note
²A rest

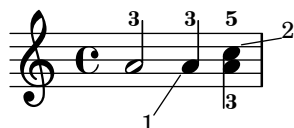
Marking a *whole* chord with an event-based footnote is not possible: a chord, even one containing just a single note, does not produce an actual event of its own. However, individual notes *inside* of the chord can be marked:

```
\book {
  \header { tagline = ##f }
  \relative c'' {
    \footnote #'(2 . 3) "Does not work" <a-3>2
```

```

<\footnote #'(-2 . -3) "Does work" a-3>4
<a-3 \footnote #'(3 . 1/2) "Also works" c-5>4
}
}

```



¹Does work
²Also works

If the footnote is to be attached to a post-event or articulation the `\footnote` command *must* be preceded by a direction indicator, `-`, `_`, `^`, and followed by the post-event or articulation to be annotated as the *music* argument. In this form the `\footnote` can be considered to be simply a copy of its last argument with a footnote mark attached to it. The syntax is:

```

direction \footnote [mark] offset footnote music
\book {
  \header { tagline = ##f }
  \relative {
    a'4_\footnote #'(0 . -1) "A slur forced down" (
      b8^\footnote #'(1 . 0.5) "A manual beam forced up" [
        b8 ]
      c4 )
      c-\footnote #'(1 . 1) "Tenuto" --
  }
}

```



¹A slur forced down
²A manual beam forced up
³Tenuto

Time-based footnotes

If the layout object being footmarked is *indirectly* caused by an event (like an `Accidental` or `Stem` caused by a `NoteHead` event), the `GrobName` of the layout object is required after the footnote text instead of *music*:

```

\book {
  \header { tagline = ##f }
  \relative c'' {

```

```

\footnote #'(-1 . -3) "A flat" Accidental
aes4 c
\footnote #'(-1 . 0.5) "Another flat" Accidental
ees
\footnote #'(1 . -2) "A stem" Stem
aes
}
}

```



¹A flat
²Another flat
³A stem

Note, however, that when a GrobName is specified, a footnote will be attached to all grobs of that type at the current time step:

```

\book {
  \header { tagline = ##f }
  \relative c' {
    \footnote #'(-1 . 3) "A flat" Accidental
    <ees ges bes>4
    \footnote #'(2 . 0.5) "Articulation" Script
    c'->-.
  }
}

```



¹A flat
²A flat
³A flat
⁴Articulation
⁵Articulation

A note inside of a chord can be given an individual (event-based) footnote. A ‘**NoteHead**’ is the only grob directly caused from a chord note, so an event-based footnote command is *only* suitable for adding a footnote to the ‘**NoteHead**’ within a chord. All other chord note grobs are indirectly caused. The `\footnote` command itself offers no syntax for specifying *both* a particular grob type *as well as* a particular event to attach to. However, one can use a time-based `\footnote` command for specifying the grob type, and then prefix this command with `\single` in order to have it applied to just the following event:

```

\book {

```



```

\header { tagline = ##f }
\relative c'' {
  < \footnote #'(1 . -2) "An A" a
    \single \footnote #'(-1 . -1) "A sharp" Accidental
    cis
    \single \footnote #'(0.5 . 0.5) "A flat" Accidental
    ees fis
  >2
}
}

```



¹A flat
²A sharp
³An A

Nota: When footnotes are attached to several musical elements at the same musical moment, as they are in the example above, the footnotes are numbered from the higher to the lower elements as they appear in the printed output, not in the order in which they are written in the input stream.

Layout objects like clefs and key-change signatures are mostly caused as a consequence of changed properties rather than actual events. Others, like bar lines and bar numbers, are a direct consequence of timing. For this reason, footnotes on such objects have to be based on their musical timing. Time-based footnotes are also preferable when marking features like stems and beams on *chords*: while such per-chord features are nominally assigned to *one* event inside the chord, relying on a particular choice would be imprudent.

The layout object in question must always be explicitly specified for time-based footnotes, and the appropriate context must be specified if the grob is created in a context other than the bottom context.

```

\book {
  \header { tagline = ##f }
  \relative c'' {
    r1 |
    \footnote #'(-0.5 . -1) "Meter change" Staff.TimeSignature
    \time 3/4
    \footnote #'(1 . -1) "Chord stem" Stem
    <c e g>4 q q
    \footnote #'(-0.5 . 1) "Bar line" Staff.BarLine
    q q
    \footnote #'(0.5 . -1) "Key change" Staff.KeySignature
    \key c \minor
    q
  }
}

```

}

¹Meter change²Chord stem³Bar line⁴Key change

Custom marks can be used as alternatives to numerical marks, and the annotation line joining the marked object to the mark can be suppressed:

```
\book {
\header { tagline = ##f }
\relative c' {
\footnote "*" #'(0.5 . -2) \markup { \italic "*" The first note" } a'4
b8
\footnote \markup { \super "$" } #'(0.5 . 1)
\markup { \super "$" \italic " The second note" } e
c4
\once \override Score.FootnoteItem.annotation-line = ##f
b-\footnote \markup \tiny "+" #'(0.1 . 0.1)
\markup { \super "+" \italic " Editorial" } \p
}
}
```

^{*} *The first note*^{\$} *The second note*⁺ *Editorial*

More examples of custom marks are shown in [Footnotes in stand-alone text], pàgina 488.

Footnotes in stand-alone text

These are for use in markup outside of music expressions. They do not have a line drawn to their point of reference: their marks simply follow the referenced markup. Marks can be inserted automatically, in which case they are numerical. Alternatively, custom marks can be provided manually.

Footnotes to stand-alone text with automatic and custom marks are created in different ways.

Footnotes in stand-alone text with automatic marks

The syntax of a footnote in stand-alone text with automatic marks is

```
\markup { ... \auto-footnote text footnote ... }
```

The elements are:

text is the markup or string to be marked.

footnote is the markup or string specifying the footnote text to use at the bottom of the page.

For example:

```
\book {
  \header { tagline = ##f }
  \markup {
    "A simple"
    \auto-footnote "tune" \italic " By me"
    "is shown below. It is a"
    \auto-footnote "recent" \italic " Aug 2012"
    "composition."
  }
  \relative {
    a'4 b8 e c4 d
  }
}
```

A simple tune¹ is shown below. It is a recent² composition.



¹ *By me*

² *Aug 2012*

Footnotes in stand-alone text with custom marks

The syntax of a footnote in stand-alone text with custom marks is

```
\markup { ... \footnote mark footnote ... }
```

The elements are:

mark is a markup or string specifying the footnote mark which is used for marking the reference point. Note that this mark is *not* inserted automatically before the footnote itself.

footnote is the markup or string specifying the footnote text to use at the bottom of the page, preceded by the *mark*.

Any easy-to-type character such as * or + may be used as a mark, as shown in [Footnotes in music expressions], pàgina 483. Alternatively, ASCII aliases may be used (see [ASCII aliases], pàgina 504):

```
\book {
  \paper { #(include-special-characters) }
  \header { tagline = ##f }
  \markup {
    "A simple tune"
    \footnote "*" \italic "*" By me"
    "is shown below. It is a recent"
    \footnote \super &dagger; \concat {
      \super &dagger; \italic " Aug 2012"
    }
    "composition."
  }
  \relative {
    a'4 b8 e c4 d
  }
}
```

A simple tune * is shown below. It is a recent [†] composition.



* *By me*

[†] *Aug 2012*

Unicode character codes may also be used to specify marks (see [Unicode], pàgina 503):

```
\book {
  \header { tagline = ##f }
  \markup {
    "A simple tune"
    \footnote \super \char##x00a7 \concat {
      \super \char##x00a7 \italic " By me"
    }
    "is shown below. It is a recent"
    \footnote \super \char##x00b6 \concat {
      \super \char##x00b6 \italic " Aug 2012"
    }
    "composition."
  }
  \relative {

```

```

    a'4 b8 e c4 d
  }
}
```

A simple tune § is shown below. It is a recent ¶ composition.



§ *By me*
 ¶ *Aug 2012*

Vegeu també

Learning Manual: Secció “Objects and interfaces” in *Manual d’aprenentatge*.

Notation Reference: [ASCII aliases], pàgina 504, [Balloon help], pàgina 225, Secció A.13 [List of special characters], pàgina 730, [Text marks], pàgina 233, [Text scripts], pàgina 230, [Unicode], pàgina 503.

Internals Reference: Secció “FootnoteEvent” in *Referència de funcionament intern*, Secció “FootnoteItem” in *Referència de funcionament intern*, Secció “FootnoteSpanner” in *Referència de funcionament intern*, Secció “Footnote-engraver” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Multiple footnotes for the same page can only be stacked, one above the other; they cannot be printed on the same line.

Footnotes cannot be attached to `MultiMeasureRests` or automatic beams or lyrics.

Footnote marks may collide with staves, `\markup` objects, other footnote marks and annotation lines.

3.2.5 Reference to page numbers

A particular place of a score can be marked using the `\label` command, either at top-level or inside music. This label can then be referred to in a markup, to get the number of the page where the marked point is placed, using the `\page-ref` markup command.

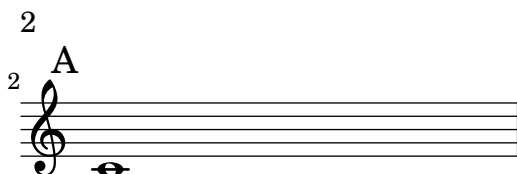
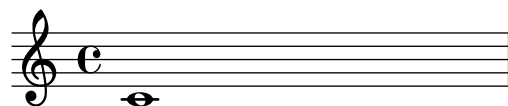
```

\header { tagline = ##f }
\book {
  \label #'firstScore
  \score {
    {
      c'1
      \pageBreak \mark A \label #'markA
      c'1
    }
  }
}
```

```

}
\markup { The first score begins on page \page-ref #'firstScore "0" "?" }
\markup { Mark A is on page \page-ref #'markA "0" "?" }
}

```



The first score begins on page 1

Mark A is on page 2

The `\page-ref` markup command takes three arguments:

1. the label, a scheme symbol, eg. `#'firstScore`;
2. a markup that will be used as a gauge to estimate the dimensions of the markup;
3. a markup that will be used in place of the page number if the label is not known;

The reason why a gauge is needed is that, at the time markups are interpreted, the page breaking has not yet occurred, so the page numbers are not yet known. To work around this issue, the actual markup interpretation is delayed to a later time; however, the dimensions of the markup have to be known before, so a gauge is used to decide these dimensions. If the book has between 10 and 99 pages, it may be "00", ie. a two digit number.

Instruccions predefinides

`\label`, `\page-ref`.

3.2.6 Table of contents

A table of contents is included using the `\markuplist \table-of-contents` command. The elements which should appear in the table of contents are entered with the `\tocItem` command, which may be used either at top-level, or inside a music expression.

```

\markuplist \table-of-contents
\pageBreak

```

```

\tocItem \markup "First score"
\score {
  {
    c'4 % ...
    \tocItem \markup "Some particular point in the first score"
    d'4 % ...
  }
}

```

```

\tocItem \markup "Second score"

```

```
\score {
  {
    e'4 % ...
  }
}
```

Markups used for formatting the table of contents are defined in the `\paper` block. There are two ‘pre-defined’ markups already available;

- `tocTitleMarkup`

Used for formatting the title of the table of contents.

```
tocTitleMarkup = \markup \huge \column {
  \fill-line { \null "Table of Contents" \null }
  \null
}
```

- `tocItemMarkup`

Used for formatting the elements within the table of contents.

```
tocItemMarkup = \markup \fill-line {
  \fromproperty #'toc:text \fromproperty #'toc:page
}
```

Both of these variables can be changed.

Here is an example changing the table of contents’ title into French;

```
\paper {
  tocTitleMarkup = \markup \huge \column {
    \fill-line { \null "Table des matières" \null }
    \hspace #1
  }
}
```

Here is an example changing the font-size of the elements in the table of contents;

```
tocItemMarkup = \markup \large \fill-line {
  \fromproperty #'toc:text \fromproperty #'toc:page
}
```

Note how the element text and page numbers are referred to in the `tocItemMarkup` definition.

The `\tocItemWithDotsMarkup` command can be included within the `tocItemMarkup` to fill the line, between a table of contents item and its corresponding page number, with dots;

```
\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

Custom commands with their own markups can also be defined to build a more complex table of contents. In the following example, a new style is defined for entering act names in a table of contents of an opera;

A new markup variable (called `tocActMarkup`) is defined in the `\paper` block;

```
\paper {
  tocActMarkup = \markup \large \column {
    \hspace #1
    \fill-line { \null \italic \fromproperty #'toc:text \null }
    \hspace #1
  }
}
```

A custom music function (`tocAct`) is then created – which uses the new `tocActMarkup` markup definition.

```
tocAct =
  #(define-music-function (text) (markup?)
    (add-toc-item! 'tocActMarkup text))
```

A LilyPond input file, using these customer definitions, could look something like this;

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

Here is an example of the `\fill-with-pattern` command used within the context of a table of contents;

```
\paper {
  tocItemMarkup = \markup { \fill-line {
    \override #'(line-width . 70)
    \fill-with-pattern #1.5 #CENTER . \fromproperty #'toc:text \fromproperty #'toc:page
  }
}
```

Vegeu també

Installed Files: `ly/toc-init.ly`.

Instruccions predefinides

`\table-of-contents`, `\tocItem`.

3.3 Working with input files

3.3.1 Including LilyPond files

A large project may be split up into separate files. To refer to another file, use

```
\include "otherfile.ly"
```

The line `\include "otherfile.ly"` is equivalent to pasting the contents of `otherfile.ly` into the current file at the place where the `\include` appears. For example, in a large project you might write separate files for each instrument part and create a “full score” file which brings together the individual instrument files. Normally the included file will define a number of variables which then become available for use in the full score file. Tagged sections can be marked in included files to assist in making them usable in different places in a score, see Secció 3.3.2 [Different editions from one source], pàgina 496.

Files in the current working directory may be referenced by specifying just the file name after the `\include` command. Files in other locations may be included by giving either a full path reference or a relative path reference (but use the UNIX forward slash, `/`, rather than the DOS/Windows back slash, `\`, as the directory separator.) For example, if `stuff.ly` is located one directory higher than the current working directory, use

```
\include "../stuff.ly"
```

or if the included orchestral parts files are all located in a subdirectory called `parts` within the current directory, use

```
\include "parts/VI.ly"
\include "parts/VII.ly"
... etc
```

Files which are to be included can also contain `\include` statements of their own. By default, these second-level `\include` statements are not interpreted until they have been brought into the main file, so the file names they specify must all be relative to the directory containing the main file, not the directory containing the included file. However, this behavior can be changed globally by passing the option `-drelative-includes` option at the command line (or by adding `#{ly:set-option 'relative-includes #t}` at the top of the main input file).

When `relative-includes` is set to `#t`, the path for each `\include` command will be taken relative to the file containing that command. This behavior is recommended and it will become the default behavior in a future version of lilypond.

Files relative to the main directory and files relative to some other directory may both be `\included` by setting `relative-includes` to `#t` or `#f` at appropriate places in the files. For example, if a general library, `libA`, has been created which itself uses sub-files which are `\included` by the entry file of that library, those `\include` statements will need to be preceded by `#{ly:set-option #relative-includes #t}` so they are interpreted correctly when brought into the main `.ly` file, like this:

```
libA/
  libA.ly
  A1.ly
  A2.ly
  ...
```

then the entry file, `libA.ly`, will contain

```
#{ly:set-option 'relative-includes #t)
\include "A1.ly"
\include "A2.ly"
...
% return to default setting
```

```
#{ly:set-option 'relative-includes #f)
```

Any .ly file can then include the entire library simply with

```
\include "~/libA/libA.ly"
```

More complex file structures may be devised by switching at appropriate places.

Files can also be included from a directory in a search path specified as an option when invoking LilyPond from the command line. The included files are then specified using just their file name. For example, to compile `main.ly` which includes files located in a subdirectory called `parts` by this method, `cd` to the directory containing `main.ly` and enter

```
lilypond --include=parts main.ly
```

and in `main.ly` write

```
\include "VI.ly"
\include "VII.ly"
... etc
```

Files which are to be included in many scores may be placed in the LilyPond directory `../ly`. (The location of this directory is installation-dependent - see Secció “Other sources of information” in *Manual d’aprenentatge*). These files can then be included simply by naming them on an `\include` statement. This is how the language-dependent files like `english.ly` are included.

LilyPond includes a number of files by default when you start the program. These includes are not apparent to the user, but the files may be identified by running `lilypond --verbose` from the command line. This will display a list of paths and files that LilyPond uses, along with much other information. Alternatively, the more important of these files are discussed in Secció “Other sources of information” in *Manual d’aprenentatge*. These files may be edited, but changes to them will be lost on installing a new version of LilyPond.

Some simple examples of using `\include` are shown in Secció “Scores and parts” in *Manual d’aprenentatge*.

Vegeu també

Learning Manual: Secció “Other sources of information” in *Manual d’aprenentatge*, Secció “Scores and parts” in *Manual d’aprenentatge*.

Advertiments i problemes coneguts

If an included file is given a name which is the same as one in LilyPond’s installation files, LilyPond’s file from the installation files takes precedence.

3.3.2 Different editions from one source

Several methods can be used to generate different versions of a score from the same music source. Variables are perhaps the most useful for combining lengthy sections of music and/or annotation. Tags are more useful for selecting one section from several alternative shorter sections of music, and can also be used for splicing pieces of music together at different points.

Whichever method is used, separating the notation from the structure of the score will make it easier to change the structure while leaving the notation untouched.

Using variables

If sections of the music are defined in variables they can be reused in different parts of the score, see Secció “Organizing pieces with variables” in *Manual d’aprenentatge*. For example, an *a cappella* vocal score frequently includes a piano reduction of the parts for rehearsal purposes which is identical to the vocal music, so the music need be entered only once. Music from two

variables may be combined on one staff, see [Automatic part combining], pàgina 178. Here is an example:

```
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 = "Tenor" {
    \clef "treble_8"
    \tenorMusic
  }
  \new Lyrics \allLyrics
  \new Staff = "Bass" {
    \clef "bass"
    \bassMusic
  }
  \new Lyrics \allLyrics
  \new PianoStaff <<
    \new Staff = "RH" {
      \partcombine \sopranoMusic \altoMusic
    }
    \new Staff = "LH" {
      \clef "bass"
      \partcombine \tenorMusic \bassMusic
    }
  >>
>>
```



Separate scores showing just the vocal parts or just the piano part can be produced by changing just the structural statements, leaving the musical notation unchanged.

For lengthy scores, the variable definitions may be placed in separate files which are then included, see Secció 3.3.1 [Including LilyPond files], pàgina 495.

Using tags

The `\tag #'partA` command marks a music expression with the name *partA*. Expressions tagged in this way can be selected or filtered out by name later, using either `\keepWithTag #'name` or `\removeWithTag #'name`. The result of applying these filters to tagged music is as follows:

Filter

Tagged music preceded by `\keepWithTag #'name` or `\keepWithTag #'(name1 name2...)`

Tagged music preceded by `\removeWithTag #'name` or `\removeWithTag #'(name1 name2...)`

Tagged music not preceded by either `\keepWithTag` or `\removeWithTag`

Result

Untagged music and music tagged with any of the given tag names is included; music tagged with any other tag name is excluded.

Untagged music and music not tagged with any of the given tag names is included; music tagged with any of the given tag names is excluded.

All tagged and untagged music is included.

The arguments of the `\tag`, `\keepWithTag` and `\removeWithTag` commands should be a symbol or list of symbols (such as `#'score` or `#'(violinI violinII)`), followed by a music expression. If *and only if* the symbols are valid LilyPond identifiers (alphabetic characters only, no numbers, underscores, or dashes) which cannot be confused with notes, the `#'` may be omitted and, as a shorthand, a list of symbols can use the dot separator: i.e., `\tag #'(violinI violinII)` can be written `\tag violinI.violinII`. The same applies to `\keepWithTag` and `\removeWithTag`.

In the following example, we see two versions of a piece of music, one showing trills with the usual notation, and one with trills explicitly expanded:

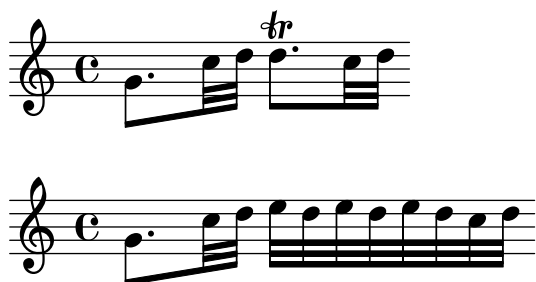
```
music = \relative {
  g'8. c32 d
```

```

\tag #'trills { d8.\trill }
\tag #'expand { \repeat unfold 3 { e32 d } }
c32 d
}

\score {
  \keepWithTag #'trills \music
}
\score {
  \keepWithTag #'expand \music
}

```



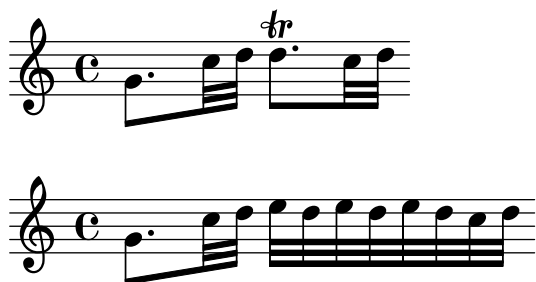
Alternatively, it is sometimes easier to exclude sections of music:

```

music = \relative {
  g'8. c32 d
  \tag #'trills { d8.\trill }
  \tag #'expand {\repeat unfold 3 { e32 d } }
  c32 d
}

\score {
  \removeWithTag #'expand
  \music
}
\score {
  \removeWithTag #'trills
  \music
}

```



Tagged filtering can be applied to articulations, texts, etc., by prepending
`-\tag #'your-tag`

to an articulation. For example, this would define a note with a conditional fingering indication and a note with a conditional annotation:

```
c1-\tag #'finger ^4
```

```
c1-\tag #'warn ^"Watch!"
```

Multiple tags may be placed on expressions with multiple `\tag` entries, or by combining multiple tags into one symbol list:

```
music = \relative c'' {
  \tag #'a \tag #'both { a4 a a a }
  \tag #'(b both) { b4 b b b }
}
<<
\keepWithTag #'a \music
\keepWithTag #'b \music
\keepWithTag #'both \music
>>
```



Multiple `\removeWithTag` filters may be applied to a single music expression to remove several differently named tagged sections. Alternatively, you can use a single `\removeWithTag` with a list of tags.

```
music = \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
  \music
  \removeWithTag #'(B C)
  \music
}
```



Using two or more `\keepWithTag` filters on a single music expression will cause *all* of the tagged sections to be removed. The first filter will remove all except the one named and any subsequent filters will remove the rest. Using one `\keepWithTag` command with a list of multiple tags will only remove tagged sections that are not specified in that list.

```
music = \relative c'' {
  \tag #'violinI { a4 a a a }
```

```

\tag #'violinII { b4 b b b }
\tag #'viola { c4 c c c }
\tag #'cello { d4 d d d }
}

\new Staff {
  \keepWithTag #'(violinI violinII)
  \music
}

```



will print `\tags violinI` and `violinII` but not `viola` or `cello`.

While `\keepWithTag` is convenient when dealing with *one* set of alternatives, the removal of music tagged with *unrelated* tags is problematic when using them for more than one purpose. In that case ‘groups’ of tags can be declared:

```
\tagGroup #'(violinI violinII viola cello)
```

Now all the different tags belong to a single ‘tag group’. Note that individual tags cannot be members of more than one *tag group*.

```
\keepWithTag #'violinI ...
```

will now only show music tagged from `violinI`’s tag group and any music tagged with one of the *other* tags will be removed.

```

music = \relative {
  \tagGroup #'(violinI violinII viola cello)
  \tag #'violinI { c''4^"violinI" c c c }
  \tag #'violinII { a2 a }
  \tag #'viola { e8 e e2. }
  \tag #'cello { d'2 d4 d }
  R1^"untagged"
}

```

```

\new Voice {
  \keepWithTag #'violinI
  \music
}

```



When using the `\keepWithTag` command, only tags from the tag groups of the tags given in the command are visible.

Sometimes you want to splice some music at a particular place in an existing music expression. You can use `\pushToTag` and `\appendToTag` for adding material at the front or end of the **elements** of an existing music construct. Not every music construct has **elements**, but sequential and simultaneous music are safe bets:

```
music = { \tag #'here { \tag #'here <<c''>> } }
```

```
{
  \pushToTag #'here c'
  \pushToTag #'here e'
  \pushToTag #'here g' \music
  \appendToTag #'here c'
  \appendToTag #'here e'
  \appendToTag #'here g' \music
}
```



Both commands get a tag, the material to splice in at every occurrence of the tag, and the tagged expression.

Vegeu també

Learning Manual: Secció “Organizing pieces with variables” in *Manual d’aprenentatge*.

Notation Reference: [Automatic part combining], pàgina 178, Secció 3.3.1 [Including LilyPond files], pàgina 495.

Advertisements i problemes coneguts

Calling `\relative` on a music expression obtained by filtering music through `\keepWithTag` or `\removeWithTag` might cause the octave relations to change, as only the pitches actually remaining in the filtered expression will be considered. Applying `\relative` first, before `\keepWithTag` or `\removeWithTag`, avoids this danger as `\relative` then acts on all the pitches as-input.

Using global settings

Global settings can be included from a separate file:

```
lilypond -dininclude-settings=MY_SETTINGS.ly MY_SCORE.ly
```

Groups of settings such as page size, font or type face can be stored in separate files. This allows different editions from the same score as well as standard settings to be applied to many scores, simply by specifying the proper settings file.

This technique also works well with the use of style sheets, as discussed in Secció “Style sheets” in *Manual d’aprenentatge*.

Vegeu també

Learning Manual: Secció “Organizing pieces with variables” in *Manual d’aprenentatge*, Secció “Style sheets” in *Manual d’aprenentatge*.

Notation Reference: Secció 3.3.1 [Including LilyPond files], pàgina 495.

3.3.3 Special characters

Text encoding

LilyPond uses the character repertoire defined by the Unicode consortium and ISO/IEC 10646. This defines a unique name and code point for the character sets used in virtually all modern languages and many others too. Unicode can be implemented using several different encodings. LilyPond uses the UTF-8 encoding (UTF stands for Unicode Transformation Format) which represents all common Latin characters in one byte, and represents other characters using a variable length format of up to four bytes.

The actual appearance of the characters is determined by the glyphs defined in the particular fonts available - a font defines the mapping of a subset of the Unicode code points to glyphs. LilyPond uses the Pango library to layout and render multi-lingual texts.

LilyPond does not perform any input-encoding conversions. This means that any text, be it title, lyric text, or musical instruction containing non-ASCII characters, must be encoded in UTF-8. The easiest way to enter such text is by using a Unicode-aware editor and saving the file with UTF-8 encoding. Most popular modern editors have UTF-8 support, for example, vim, Emacs, jEdit, and Gedit do. All MS Windows systems later than NT use Unicode as their native character encoding, so even Notepad can edit and save a file in UTF-8 format. A more functional alternative for Windows is BabelPad.

If a LilyPond input file containing a non-ASCII character is not saved in UTF-8 format the error message

```
FT_Get_Glyph_Name () error: invalid argument
```

will be generated.

Here is an example showing Cyrillic, Hebrew and Portuguese text:



Unicode

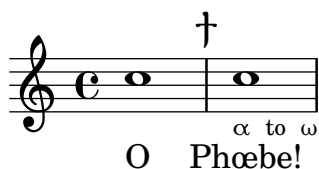
To enter a single character for which the Unicode code point is known but which is not available in the editor being used, use either `\char ##xhhhh` or `\char #dddd` within a `\markup` block, where `hhhh` is the hexadecimal code for the character required and `dddd` is the corresponding decimal value. Leading zeroes may be omitted, but it is usual to specify all four characters in the hexadecimal representation. (Note that the UTF-8 encoding of the code point should *not* be used after `\char`, as UTF-8 encodings contain extra bits indicating the number of octets.) Unicode code charts and a character name index giving the code point in hexadecimal for any character can be found on the Unicode Consortium website, <http://www.unicode.org/>.

For example, `\char ##x03BE` and `\char #958` would both enter the Unicode U+03BE character, which has the Unicode name “Greek Small Letter Xi”.

Any Unicode code point may be entered in this way and if all special characters are entered in this format it is not necessary to save the input file in UTF-8 format. Of course, a font containing all such encoded characters must be installed and available to LilyPond.

The following example shows Unicode hexadecimal values being entered in four places – in a rehearsal mark, as articulation text, in lyrics and as stand-alone text below the score:

```
\score {
  \relative {
    c''1 \mark \markup { \char ##x03EE }
    c1_\markup { \tiny { \char ##x03B1 " to " \char ##x03C9 } }
  }
  \addlyrics { 0 \markup { \concat { Ph \char ##x0153 be! } } }
}
\markup { "Copyright 2008--2015" \char ##x00A9 }
```



Copyright 2008--2015 ©

To enter the copyright sign in the copyright notice use:

```
\header {
  copyright = \markup { \char ##x00A9 "2008" }
}
```

ASCII aliases

A list of ASCII aliases for special characters can be included:

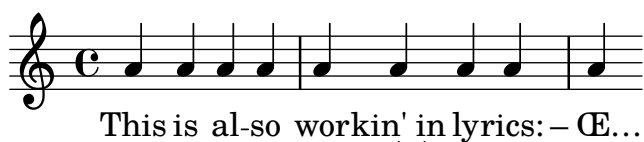
```
\paper {
  #(include-special-characters)
}
```

```
\markup "&flqq; &ndash; &OE;uvre incomplète&hellip;&frqq;"
```

```
\score {
  \new Staff { \repeat unfold 9 a'4 }
  \addlyrics {
    This is al -- so wor -- kin'~in ly -- rics: &ndash;_&OE;&hellip;
  }
}
```

```
\markup \column {
  "The replacement can be disabled:"
  "&ndash; &OE; &hellip;"
  \override #'(replacement-alist . ()) "&ndash; &OE; &hellip;"
}
```

« – Œuvre incomplète... »



The replacement can be disabled:

– Œ ...

– &OE; …

You can also make your own aliases, either globally:

```
\paper {
  #(add-text-replacements!
    '(("100" . "hundred")
      ("dpi" . "dots per inch")))
}
\markup "A 100 dpi."
```

A hundred dots per inch.

or locally:

```
\markup \replace #'(("100" . "hundred")
                      ("dpi" . "dots per inch")) "A 100 dpi."
```

A hundred dots per inch.

Vegeu també

Notation Reference: Secció A.13 [List of special characters], pàgina 730.

Installed Files: `ly/text-replacements.ly`.

3.4 Controlling output

3.4.1 Extracting fragments of music

It is possible to output one or more fragments of a score by defining the explicit location of the music to be extracted within the `\layout` block of the input file using the `clip-regions` function, and then running LilyPond with the `-dclip-systems` option;

```
\layout {
  clip-regions
  = #(list
    (cons
      (make-rhythmic-location 5 1 2)
      (make-rhythmic-location 7 3 4)))
}
```

This example will extract a single fragment of the input file *starting* after a half-note duration in fifth measure (5 1 2) and *ending* after the third quarter-note in the seventh measure (7 3 4).

Additional fragments can be extracted by adding more pairs of `make-rhythmic-location` entries to the `clip-regions` list in the `\layout` block.

By default, each music fragment will be output as a separate EPS file, but other formats such as PDF or PNG can also be created if required. The extracted music is output as if had been literally ‘cut’ from the original printed score so if a fragment runs over one or more lines, a separate output file for each line will be generated.

Vegeu també

Notation Reference: Secció 4.2.1 [The layout block], pàgina 531.

Application Usage: Secció “Command-line usage” in *Utilització del programa*.

3.4.2 Skipping corrected music

When entering or copying music, usually only the music near the end (where you are adding notes) is interesting to view and correct. To speed up this correction process, it is possible to skip typesetting of all but the last few measures. This is achieved by putting

```
showLastLength = R1*5
\score { ... }
```

in your source file. This will render only the last 5 measures (assuming 4/4 time signature) of every `\score` in the input file. For longer pieces, rendering only a small part is often an order of magnitude quicker than rendering it completely. When working on the beginning of a score you have already typeset (e.g., to add a new part), the `showFirstLength` property may be useful as well.

Skipping parts of a score can be controlled in a more fine-grained fashion with the property `Score.skipTypesetting`. When it is set, no typesetting is performed at all.

This property is also used to control output to the MIDI file. Note that it skips all events, including tempo and instrument changes. You have been warned.

```
\relative c' {
  c1
  \set Score.skipTypesetting = ##t
  \tempo 4 = 80
  c4 c c c
  \set Score.skipTypesetting = ##f
  d4 d d d
}
```



In polyphonic music, `Score.skipTypesetting` will affect all voices and staves, saving even more time.

3.4.3 Alternative output formats

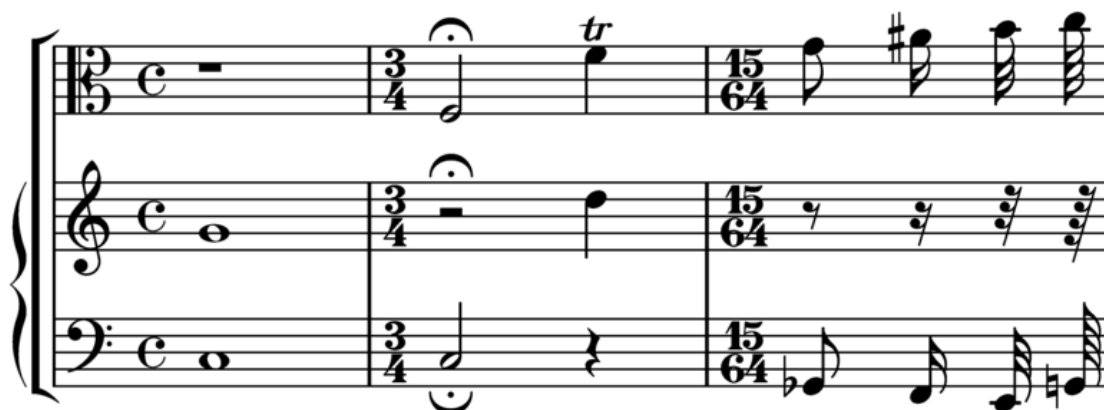
The default output formats for the printed score are Portable Document Format (PDF) and PostScript (PS). Portable Network Graphics (PNG), Scalable Vector Graphics (SVG) and Encapsulated PostScript (EPS) output is available through the command line option, see Secció “Basic command line options for LilyPond” in *Utilització del programa*.

3.4.4 Replacing the notation font

Gonville is an alternative to the Feta font used in LilyPond and can be downloaded from:

<http://www.chiark.greenend.org.uk/~sgtatham/gonville/> (<http://www.chiark.greenend.org.uk/~sgtatham/gonville/>)

Here are a few sample bars of music set in Gonville:



Here are a few sample bars of music set in LilyPond’s Feta font:



Installation Instructions for MacOS

Download and extract the zip file. Copy the `lilyfonts` directory to `SHARE_DIR/lilypond/current`; for more information, see Secció “Other sources of information” in *Manual d’aprenentatge*. Rename the existing `fonts` directory to `fonts_orig` and the `lilyfonts` directory to `fonts`. To revert back to Feta, reverse the process.

Vegeu també

Learning Manual: Secció “Other sources of information” in *Manual d’aprenentatge*.

Advertiments i problemes coneguts

Gonville cannot be used to typeset ‘Ancient Music’ notation and it is likely newer glyphs in later releases of LilyPond may not exist in the Gonville font family. Please refer to the author’s website for more information on these and other specifics, including licensing of Gonville.

3.5 Creating MIDI output

LilyPond can produce files that conform to the MIDI (Musical Instrument Digital Interface) standard and so allow for the checking of the music output aurally (with the help of an application or device that understands MIDI). Listening to MIDI output may also help in spotting errors such as notes that have been entered incorrectly or are missing accidentals and so on.

MIDI files do not contain sound (like AAC, MP3 or Vorbis files) but require additional software to produce sound from them.

3.5.1 Supported notation for MIDI

The following musical notation can be used with LilyPond’s default capabilities to produce MIDI output;

- Breath marks
- Chords entered as chord names
- Crescendi, decrescendi over multiple notes. The volume is altered linearly between the two extremes
- Dynamic markings from `ppppp` to `fffff`, including `mp`, `mf` and `sf`
- Microtones but *not* microtonal chords. A MIDI player that supports pitch bending will also be required.
- Lyrics

- Pitches
- Rhythms entered as note durations, including tuplets
- ‘Simple’ articulations; staccato, staccatissimo, accent, marcato and portato
- Tempo changes using the `\tempo` function
- Ties
- Tremolos that are *not* entered with a ‘:[*number*]’ value

Panning, balance, expression, reverb and chorus effects can also be controlled by setting context properties, see Secció 3.5.8 [Context properties for MIDI effects], pàgina 517.

When combined with the `articulate` script the following, additional musical notation can be output to MIDI;

- Appoggiaturas. These are made to take half the value of the note following (without taking dots into account). For example;

```
\appoggiatura c8 d2.
```

The `c` will take the value of a crotchet.

- Ornaments (i.e., mordents, trills and turns et al.)
- Rallentando, accelerando, ritardando and a tempo
- Slurs, including phrasing slurs
- Tenuto

See Secció 3.5.9 [Enhancing MIDI output], pàgina 518.

3.5.2 Unsupported notation for MIDI

The following items of musical notation cannot be output to MIDI;

- Articulations other than staccato, staccatissimo, accent, marcato and portato
- Crescendi and decrescendi over a *single* note
- Fermata
- Figured bass
- Glissandi
- Falls and doits
- Microtonal chords
- Rhythms entered as annotations, e.g., swing
- Tempo changes without `\tempo` (e.g., entered as annotations)
- Tremolos that *are* entered with a ‘:[*number*]’ value

3.5.3 The MIDI block

To create a MIDI output file from a LilyPond input file, insert a `\midi` block, which can be empty, within the `\score` block;

```
\score {
  ... music ...
  \layout { }
  \midi { }
}
```

Nota: A `\score` block that, as well as the music, contains only a `\midi` block (i.e., *without* the `\layout` block), will only produce MIDI output files. No notation will be printed.

The default output file extension (`.midi`) can be changed by using the `-dmidi-extension` option with the `lilypond` command:

```
lilypond -dmidi-extension=mid MyFile.ly
```

Alternatively, add the following Scheme expression before the start of either the `\book`, `\bookpart` or `\score` blocks. See Secció 3.1.5 [File structure], pàgina 468.

```
#{ly:set-option 'midi-extension "mid"}
```

Vegeu també

Notation Reference: Secció 3.1.5 [File structure], pàgina 468, Secció 3.2.3 [Creating output file metadata], pàgina 482.

Installed Files: `scm/midi.scm`.

Advertiments i problemes coneguts

There are fifteen MIDI channels available and one additional channel (`#10`) for drums. Staves are assigned to channels in sequence, so a score that contains more than fifteen staves will result in the extra staves sharing (but not overwriting) the same MIDI channel. This may be a problem if the sharing staves have conflicting, channel-based, MIDI properties – such as different MIDI instruments – set.

3.5.4 Controlling MIDI dynamics

It is possible to control the overall MIDI volume, the relative volume of dynamic markings and the relative volume of different instruments.

Dynamic marks translate automatically into volume levels in the available MIDI volume range whereas crescendi and decrescendi vary the volume linearly between their two extremes. It is possible to control the relative volume of dynamic markings, and the overall volume levels of different instruments.

Dynamic marks in MIDI

Only the dynamic markings from `ppppp` to `fffff`, including `mp`, `mf` and `sf` have values assigned to them. This value is then applied to the value of the overall MIDI volume range to obtain the final volume included in the MIDI output for that particular dynamic marking. The default fractions range from 0.25 for `ppppp` to 0.95 for `fffff`. The complete set of dynamic marks and their associated fractions can be found in `scm/midi.scm`.

Fragments de codi seleccionats

Creating custom dynamics in MIDI output

The following example shows how to create a dynamic marking, not included in the default list, and assign it a specific value so that it can be used to affect MIDI output.

The dynamic mark `\rfz` is assigned a value of 0.9.

```

(define (myDynamics dynamic)
  (if (equal? dynamic "rfz")
      0.9
      (default-dynamic-absolute-volume dynamic)))

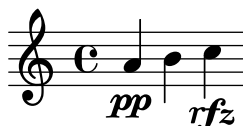
\score {
  \new Staff {
    \set Staff.midiInstrument = #"cello"
    \set Score.dynamicAbsoluteVolumeFunction = #myDynamics
    \new Voice {

```

```

        \relative {
          a'4\pp b c-\rfz
        }
      }
    }
  \layout {}
  \midi {}
}

```



Installed Files: `ly/script-init.ly scm/midi.scm`.

Snippets: Secció “MIDI” in *Fragments de codi*.

Internals Reference: Secció “Dynamic_performer” in *Referència de funcionament intern*.

Setting MIDI volume

The minimum and maximum overall volume of MIDI dynamic markings is controlled by setting the properties `midiMinimumVolume` and `midiMaximumVolume` at the `Score` level. These properties have an effect only at the start of a voice and on dynamic marks. The fraction corresponding to each dynamic mark is modified with this formula

$$\text{midiMinimumVolume} + (\text{midiMaximumVolume} - \text{midiMinimumVolume}) * \text{fraction}$$

In the following example the dynamic range of the overall MIDI volume is limited to the range 0.2 - 0.5.

```

\score {
  <<
    \new Staff {
      \set Staff.midiInstrument = #"flute"
      ... music ...
    }
    \new Staff {
      \set Staff.midiInstrument = #"clarinet"
      ... music ...
    }
  >>
  \midi {
    \context {
      \Score
      midiMinimumVolume = #0.2
      midiMaximumVolume = #0.5
    }
  }
}

```

Simple MIDI instrument equalization can be achieved by setting `midiMinimumVolume` and `midiMaximumVolume` properties within the `Staff` context.

```

\score {
  \new Staff {
    \set Staff.midiInstrument = #"flute"
    \set Staff.midiMinimumVolume = #0.7

```



```

    \set Staff.midiMaximumVolume = #0.9
    ... music ...
  }
  \midi { }
}

```

For scores with multiple staves and multiple MIDI instruments, the relative volumes of each instrument can be set individually;

```

\score {
  <<
    \new Staff {
      \set Staff.midiInstrument = #"flute"
      \set Staff.midiMinimumVolume = #0.7
      \set Staff.midiMaximumVolume = #0.9
      ... music ...
    }
    \new Staff {
      \set Staff.midiInstrument = #"clarinet"
      \set Staff.midiMinimumVolume = #0.3
      \set Staff.midiMaximumVolume = #0.6
      ... music ...
    }
  >>
  \midi { }
}

```

In this example the volume of the clarinet is reduced relative to the volume of the flute.

If these volumes properties are not set then LilyPond still applies a ‘small degree’ of equalization to certain instruments. See `scm/midi.scm`.

Installed Files: `scm/midi.scm`.

Vegeu també

Notation Reference: Secció 4.2 [Score layout], pàgina 531.

Internals Reference: Secció “Dynamic_performer” in *Referència de funcionament intern*.

Fragments de codi seleccionats

Replacing default MIDI instrument equalization

The default MIDI instrument equalizer can be replaced by setting the `instrumentEqualizer` property in the `Score` context to a user-defined Scheme procedure that uses a MIDI instrument name as its argument along with a pair of fractions indicating the minimum and maximum volumes respectively to be applied to that specific instrument.

The following example sets the minimum and maximum volumes for flute and clarinet respectively.

```

#(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 { }
}

```



Advertiments i problemes coneguts

Changes in the MIDI volume take place only on starting a note, so crescendi and decrescendi cannot affect the volume of a single note.

Setting MIDI block properties

The `\midi` block can contain context rearrangements, new context definitions or code that sets the values of certain properties.

```

\score {
  ... music ...
  \midi {
    \tempo 4 = 72
  }
}

```

```
}
}
```

Here the tempo is set to 72 quarter-note beats per minute. The tempo mark in the `\midi` block will not appear in the printed score. Although any other `\tempo` indications specified within the `\score` block will also be reflected in the MIDI output.

In a `\midi` block the `\tempo` command is setting properties during the interpretation of the music and in the context of output definitions; so it is interpreted *as if* it were a context modification.

Context definitions follow the same syntax as those in a `\layout` block;

```
\score {
  ... music ...
  \midi {
    \context {
      \Voice
      \remove "Dynamic_performer"
    }
  }
}
```

This example removes the effect of dynamics from the MIDI output. Note: LilyPond's translation modules used for sound are called 'performers'.

Vegeu també

Learning Manual: Secció "Other sources of information" in *Manual d'aprenentatge*.

Notation Reference: Secció 1.3 [Expressive marks], pàgina 119, Secció 4.2 [Score layout], pàgina 531.

Installed Files: `ly/performer-init.ly`.

Snippets: Secció "MIDI" in *Fragments de codi*.

Internals Reference: Secció "Dynamic_performer" in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Some MIDI players do not always correctly handle tempo changes in the midi output.

Changes to the `midiInstrument`, as well as some MIDI options, at the *beginning* of a staff may appear twice in the MIDI output.

3.5.5 Using MIDI instruments

MIDI instruments are set using the `midiInstrument` property within a `Staff` context.

```
\score {
  \new Staff {
    \set Staff.midiInstrument = #"glockenspiel"
    ... music ...
  }
  \midi { }
}

or

\score {
  \new Staff \with {midiInstrument = #"cello"} {
    ... music ...
  }
}
```

```
\midi { }
}
```

If the instrument name does not match any of the instruments listed in the ‘MIDI instruments’ section, the **acoustic grand** instrument will be used instead. See Secció A.6 [MIDI instruments], pàgina 649.

Vegeu també

Learning Manual: Secció “Other sources of information” in *Manual d’aprenentatge*.

Notation Reference: Secció A.6 [MIDI instruments], pàgina 649, Secció 4.2 [Score layout], pàgina 531.

Installed Files: `scm/midi.scm`.

Advertiments i problemes coneguts

Percussion instruments that are notated in a **DrumStaff** context will be output, correctly, to MIDI channel 10 but some pitched, percussion instruments like the xylophone, marimba, vibraphone or timpani, are treated as “normal” instruments so the music for these should be entered in a **Staff** (not **DrumStaff**) context to obtain correct MIDI output. A full list of **channel 10 drum-kits** entries can be found in `scm/midi.scm`. See Secció “Other sources of information” in *Manual d’aprenentatge*.

3.5.6 Using repeats with MIDI

Repeats can be represented in the MIDI output by applying the `\unfoldRepeats` command.

```
\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 { }
}
```

In order to restrict the effect of `\unfoldRepeats` to the MIDI output only, while also generating printable scores, it is necessary to make *two* `\score` blocks; one for MIDI (with unfolded repeats) and one for the notation (with volta, tremolo, and percent repeats);

```
\score {
  ... music ...
  \layout { }
}
\score {
  \unfoldRepeats {
    ... music ...
  }
  \midi { }
}
```

When using multiple voices, each of the voices must contain completely unfolded repeats for correct MIDI output.

Vegeu també

Notation Reference: Secció 1.4 [Repeats], pàgina 147.

3.5.7 MIDI channel mapping

When generating a MIDI file from a score, LilyPond will automatically assign every note in the score to a MIDI channel, the one on which it should be played when it is sent to a MIDI device. A MIDI channel has a number of controls available to select, for example, the instrument to be used to play the notes on that channel, or to request the MIDI device to apply various effects to the sound produced on the channel. At all times, every control on a MIDI channel can have only a single value assigned to it (which can be modified, however, for example, to switch to another instrument in the middle of a score).

The MIDI standard supports only 16 channels per MIDI device. This limit on the number of channels also limits the number of different instruments which can be played at the same time.

LilyPond creates separate MIDI tracks for each staff, (or discrete instrument or voice, depending on the value of `Score.midiChannelMapping`), and also for each lyrics context. There is no limit to the number of tracks.

To work around the limited number of MIDI channels, LilyPond supports a number of different modes for MIDI channel allocation, selected using the `Score.midiChannelMapping` context property. In each case, if more MIDI channels than the limit are required, the allocated channel numbers wrap around back to 0, possibly causing the incorrect assignment of instruments to some notes. This context property can be set to one of the following values:

'staff

Allocate a separate MIDI channel to each staff in the score (this is the default). All notes in all voices contained within each staff will share the MIDI channel of their enclosing staff, and all are encoded in the same MIDI track.

The limit of 16 channels is applied to the total number of staff and lyrics contexts, even though MIDI lyrics do not take up a MIDI channel.

'instrument

Allocate a separate MIDI channel to each distinct MIDI instrument specified in the score. This means that all the notes played with the same MIDI instrument will share the same MIDI channel (and track), even if the notes come from different voices or staves.

In this case the lyrics contexts do not count towards the MIDI channel limit of 16 (as they will not be assigned to a MIDI instrument), so this setting may allow a better allocation of MIDI channels when the number of staves and lyrics contexts in a score exceeds 16.

'voice

Allocate a separate MIDI channel to each voice in the score that has a unique name among the voices in its enclosing staff. Voices in different staves are always assigned separate MIDI channels, but any two voices contained within the same staff will share the same MIDI channel if they have the same name. Because `midiInstrument` and the several MIDI controls for effects are properties of the staff context, they cannot be set separately for each voice. The first voice will be played with the instrument and effects specified for the staff, and voices with a different name from the first will be assigned the default instrument and effects.

Note: different instruments and/or effects can be assigned to several voices on the same staff by moving the `Staff_performer` from the `Staff` to the `Voice` context, and leaving `midiChannelMapping` to default to `'staff` or set to `'instrument`; see the snippet below.

For example, the default MIDI channel mapping of a score can be changed to the 'instrument setting as shown:

```
\score {
  ...music...
  \midi {
    \context {
      \Score
      midiChannelMapping = #'instrument
    }
  }
}
```

Fragments de codi seleccionats

Changing MIDI output to one channel per voice

When outputting MIDI, the default behavior is for each staff to represent one MIDI channel, with all the voices on a staff amalgamated. This minimizes the risk of running out of MIDI channels, since there are only 16 available per track.

However, by moving the `Staff_performer` to the `Voice` context, each voice on a staff can have its own MIDI channel, as is demonstrated by the following example: despite being on the same staff, two MIDI channels are created, each with a different `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 Context properties for MIDI effects

The following context properties can be used to apply various MIDI effects to notes played on the MIDI channel associated with the current staff, MIDI instrument or voice (depending on the value of the `Score.midiChannelMapping` context property and the context in which the `Staff_performer` is located; see Secció 3.5.7 [MIDI channel mapping], pàgina 515).

Changing these context properties will affect all notes played on the channel after the change, however some of the effects may even apply also to notes which are already playing (depending on the implementation of the MIDI output device).

The following context properties are supported:

`Staff.midiPanPosition`

The pan position controls how the sound on a MIDI channel is distributed between left and right stereo outputs. The context property accepts a number between -1.0 (`#LEFT`) and 1.0 (`#RIGHT`); the value -1.0 will put all sound power to the left stereo output (keeping the right output silent), the value 0.0 (`#CENTER`) will distribute the sound evenly between the left and right stereo outputs, and the value 1.0 will move all sound to the right stereo output. Values between -1.0 and 1.0 can be used to obtain mixed distributions between left and right stereo outputs.

`Staff.midiBalance`

The stereo balance of a MIDI channel. Similarly to the pan position, this context property accepts a number between -1.0 (`#LEFT`) and 1.0 (`#RIGHT`). It varies the relative volume sent to the two stereo speakers without affecting the distribution of the stereo signals.

`Staff.midiExpression`

Expression level (as a fraction of the maximum available level) to apply to a MIDI channel. A MIDI device combines the MIDI channel's expression level with a voice's current dynamic level (controlled using constructs such as `\p` or `\ff`) to obtain the total volume of each note within the voice. The expression control could be used, for example, to implement crescendo or decrescendo effects over single sustained notes (not supported automatically by LilyPond).

The expression level ranges from 0.0 (no expression, meaning zero volume) to 1.0 (full expression).

`Staff.midiReverbLevel`

Reverb level (as a fraction of the maximum available level) to apply to a MIDI channel. This property accepts numbers between 0.0 (no reverb) and 1.0 (full effect).

`Staff.midiChorusLevel`

Chorus level (as a fraction of the maximum available level) to apply to a MIDI channel. This property accepts numbers between 0.0 (no chorus effect) and 1.0 (full effect).

Advertiments i problemes coneguts

As MIDI files do not contain any actual audio data, changes in these context properties translate only to requests for changing MIDI channel controls in the outputted MIDI files. Whether a particular MIDI device (such as a software MIDI player) can actually handle any of these requests in a MIDI file is entirely up to the implementation of the device: a device may choose to ignore some or all of these requests. Also, how a MIDI device will interpret different values for these controls (generally, the MIDI standard fixes the behavior only at the endpoints of the value range available for each control), and whether a change in the value of a control will affect notes already playing on that MIDI channel or not, is also specific to the MIDI device implementation.

When generating MIDI files, LilyPond will simply transform the fractional values within each range linearly into values in a corresponding (7-bit, or 14-bit for MIDI channel controls which support fine resolution) integer range (0-127 or 0-32767, respectively), rounding fractional values towards the nearest integer away from zero. The converted integer values are stored as-is in the generated MIDI file. Please consult the documentation of your MIDI device for information about how the device interprets these values.

3.5.9 Enhancing MIDI output

The default MIDI output is basic but can be improved by setting MIDI instruments, `\midi` block properties and/or using the `articulate` script.

The `articulate` script

To use the `articulate` script add the appropriate `\include` command at the top of the input file;

```
\include "articulate.ly"
```

The script creates MIDI output into appropriately ‘time-scaled’ notes to match many articulation and tempo indications. Engraved output however, will also be altered to literally match the MIDI output.

```
\score {
  \articulate <<
    ... music ...
  >>
  \midi { }
}
```

The `\articulate` command enables abbreviations (such as trills and turns) to be processed. A full list of supported items can be found in the script itself. See `ly/articulate.ly`.

Vegeu també

Learning Manual: Secció “Other sources of information” in *Manual d’aprenentatge*.

Notation Reference: Secció 4.2 [Score layout], pàgina 531.

Installed Files: `ly/articulate.ly`.

Nota: The `articulate` script may shorten chords, which might not be appropriate for some types of instrument, such as organ music. Notes that do not have any articulations attached to them may also be shortened; so to allow for this, restrict the use of the `\articulate` function to shorter segments of music, or modify the values of the variables defined in the `articulate` script to compensate for the note-shortening behavior.

3.6 Extracting musical information

In addition to creating graphical output and MIDI, LilyPond can display musical information as text.

3.6.1 Displaying LilyPond notation

Displaying a music expression in LilyPond notation can be done with the music function `\displayLilyMusic`. To see the output, you will typically want to call LilyPond using the command line. For example,

```
{
  \displayLilyMusic \transpose c a, { c4 e g a bes }
}

will display

{ a,4 cis4 e4 fis4 g4 }
```

By default, LilyPond will print these messages to the console along with all the other LilyPond compilation messages. To split up these messages and save the results of `\displayLilyMusic`, redirect the output to a file.

```
lilypond file.ly >display.txt
```

Note that LilyPond does not just display the music expression, but also interprets it (since `\displayLilyMusic` returns it in addition to displaying it). Just insert `\displayLilyMusic` into the existing music in order to get information about it.

To interpret and display a music section in the console but, at the same time, remove it from the output file use the `\void` command.

```
{
  \void \displayLilyMusic \transpose c a, { c4 e g a bes }
  c1
}
```

3.6.2 Displaying scheme music expressions

See Secció “Displaying music expressions” in *Extendre*.

3.6.3 Saving music events to a file

Music events can be saved to a file on a per-staff basis by including a file in your main score.

```
\include "event-listener.ly"
```

This will create file(s) called `FILENAME-STAFFNAME.notes` or `FILENAME-unnamed-staff.notes` for each staff. Note that if you have multiple unnamed staves, the events for all staves will be mixed together in the same file. The output looks like this:

```
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
```

The syntax is a tab-delimited line, with two fixed fields on each line followed by optional parameters.

```
time  type  ...params...
```

This information can easily be read into other programs such as python scripts, and can be very useful for researchers wishing to perform musical analysis or playback experiments with LilyPond.

Advertiments i problemes coneguts

Not all lilypond music events are supported by `event-listener.ly`. It is intended to be a well-crafted “proof of concept”. If some events that you want to see are not included, copy `event-listener.ly` into your lilypond directory and modify the file so that it outputs the information you want.

4 Spacing issues

The global paper layout is determined by three factors: the page layout, the line breaks, and the spacing. These all influence each other. The choice of spacing determines how densely each system of music is set. This influences where line breaks are chosen, and thus ultimately, how many pages a piece of music takes.

Globally speaking, this procedure happens in four steps: first, flexible distances (‘springs’) are chosen, based on durations. All possible line breaking combinations are tried, and a ‘badness’ score is calculated for each. Then the height of each possible system is estimated. Finally, a page breaking and line breaking combination is chosen so that neither the horizontal nor the vertical spacing is too cramped or stretched.

Two types of blocks can contain layout settings: `\paper {...}` and `\layout {...}`. The `\paper` block contains page layout settings that are expected to be the same for all scores in a book or bookpart, such as the paper height, or whether to print page numbers, etc. See Secció 4.1 [Page layout], pàgina 521. The `\layout` block contains score layout settings, such as the number of systems to use, or the space between staff-groups, etc. See Secció 4.2 [Score layout], pàgina 531.

4.1 Page layout

This section discusses page layout options for the `\paper` block.

4.1.1 The `\paper` block

`\paper` blocks may be placed in three different places to form a descending hierarchy of `\paper` blocks:

- At the top of the input file, before all `\book`, `\bookpart`, and `\score` blocks.
- Within a `\book` block but outside all the `\bookpart` and `\score` blocks within that book.
- Within a `\bookpart` block but outside all `\score` blocks within that bookpart.

A `\paper` block cannot be placed within a `\score` block.

The values of the fields filter down this hierarchy, with the values set higher in the hierarchy persisting unless they are over-ridden by a value set lower in the hierarchy.

Several `\paper` blocks can appear at each of the levels, for example as parts of several `\included` files. If so, the fields at each level are merged, with values encountered last taking precedence if duplicated fields appear.

Settings that can appear in a `\paper` block include:

- the `set-paper-size` scheme function,
- `\paper` variables used for customizing page layout, and
- markup definitions used for customizing the layout of headers, footers, and titles.

The `set-paper-size` function is discussed in the next section, Secció 4.1.2 [Paper size and automatic scaling], pàgina 522. The `\paper` variables that deal with page layout are discussed in later sections. The markup definitions that deal with headers, footers, and titles are discussed in Secció 3.2.2 [Custom titles headers and footers], pàgina 477.

Most `\paper` variables will only work in a `\paper` block. The few that will also work in a `\layout` block are listed in Secció 4.2.1 [The `\layout` block], pàgina 531.

Except when specified otherwise, all `\paper` variables that correspond to distances on the page are measured in millimeters, unless a different unit is specified by the user. For example, the following declaration sets `top-margin` to ten millimeters:

```
\paper {
```

```
top-margin = 10
}
```

To set it to 0.5 inches, use the `\in` unit suffix:

```
\paper {
  top-margin = 0.5\in
}
```

The available unit suffixes are `\mm`, `\cm`, `\in`, and `\pt`. These units are simple values for converting from millimeters; they are defined in `ly/paper-defaults-init.ly`. For the sake of clarity, when using millimeters, the `\mm` is typically included in the code, even though it is not technically necessary.

It is also possible to define `\paper` values using Scheme. The Scheme equivalent of the above example is:

```
\paper {
  #(define top-margin (* 0.5 in))
}
```

Vegeu també

Notation Reference: Secció 4.1.2 [Paper size and automatic scaling], pàgina 522, Secció 3.2.2 [Custom titles headers and footers], pàgina 477, Secció 4.2.1 [The `\layout` block], pàgina 531.

Installed Files: `ly/paper-defaults-init.ly`.

4.1.2 Paper size and automatic scaling

Setting the paper size

‘A4’ is the default value when no explicit paper size is set. However, there are two functions that can be used to change it:

```
set-default-paper-size
  #(set-default-paper-size "quarto")
  which must always be placed at the toplevel scope, and

set-paper-size
  \paper {
    #(set-paper-size "tabloid")
  }
  which must always be placed in a \paper block.
```

If the `set-default-paper-size` function is used in the toplevel scope, it must come before any `\paper` block. `set-default-paper-size` sets the paper size for all pages, whereas `set-paper-size` only sets the paper size for the pages that the `\paper` block applies to. For example, if the `\paper` block is at the top of the file, then it will apply the paper size to all pages. If the `\paper` block is inside a `\book`, then the paper size will only apply to that book.

When the `set-paper-size` function is used, it must be placed *before* any other functions used within the same `\paper` block. See [Automatic scaling to paper size], pàgina 523.

Paper sizes are defined in `scm/paper.scm`, and while it is possible to add custom sizes, they will be overwritten on subsequent software updates. The available paper sizes are listed in Secció A.5 [Predefined paper sizes], pàgina 645.

The following command can be used in the file to add a custom paper size which can then be used with `set-default-paper-size` or `set-paper-size` as appropriate,

```
#(set! paper-alist (cons '("my size" . (cons (* 15 in) (* 3 in))) paper-alist))
```

```
\paper {
  #(set-paper-size "my size")
}
```

The units `in` (inches), `cm` (centimeters) and `mm` (millimeters) can all be used.

If the symbol `'landscape'` is added to the paper size function, pages will be rotated by 90 degrees, and wider line widths will be set accordingly.

```
#(set-default-paper-size "a6" 'landscape)
```

Swapping the paper dimensions *without* having the print rotated (like when printing to postcard size, or creating graphics for inclusion rather than a standalone document) can be achieved by appending `'landscape'` to the name of the paper size itself:

```
#(set-default-paper-size "a6landscape")
```

When the paper size ends with an explicit `'landscape'` or `'portrait'`, the presence of a `'landscape'` symbol *only* affects print orientation, not the paper dimensions used for layout.

Vegeu també

Notation Reference: [Automatic scaling to paper size], pàgina 523, Secció A.5 [Predefined paper sizes], pàgina 645.

Installed Files: `scm/paper.scm`.

Automatic scaling to paper size

If the paper size is changed with one of the scheme functions (`set-default-paper-size` or `set-paper-size`), the values of several `\paper` variables are automatically scaled to the new size. To bypass the automatic scaling for a particular variable, set the variable after setting the paper size. Note that the automatic scaling is not triggered by setting the `paper-height` or `paper-width` variables, even though `paper-width` can influence other values (this is separate from scaling and is discussed below). The `set-default-paper-size` and `set-paper-size` functions are described in [Setting the paper size], pàgina 522.

The vertical dimensions affected by automatic scaling are `top-margin` and `bottom-margin` (see Secció 4.1.3 [Fixed vertical spacing `\paper` variables], pàgina 523). The horizontal dimensions affected by automatic scaling are `left-margin`, `right-margin`, `inner-margin`, `outer-margin`, `binding-offset`, `indent`, and `short-indent` (see Secció 4.1.5 [Horizontal spacing `\paper` variables], pàgina 526).

The default values for these dimensions are set in `ly/paper-defaults-init.ly`, using internal variables named `top-margin-default`, `bottom-margin-default`, etc. These are the values that result at the default paper size `a4`. For reference, with `a4` paper the `paper-height` is `297\mm` and the `paper-width` is `210\mm`.

Vegeu també

Notation Reference: Secció 4.1.3 [Fixed vertical spacing `\paper` variables], pàgina 523, Secció 4.1.5 [Horizontal spacing `\paper` variables], pàgina 526.

Installed Files: `ly/paper-defaults-init.ly`, `scm/paper.scm`.

4.1.3 Fixed vertical spacing `\paper` variables

Nota: Some `\paper` dimensions are automatically scaled to the paper size, which may lead to unexpected behavior. See [Automatic scaling to paper size], pàgina 523.

Default values (before scaling) are defined in `ly/paper-defaults-init.ly`.

paper-height

The height of the page, unset by default. Note that the automatic scaling of some vertical dimensions is not affected by this.

top-margin

The margin between the top of the page and the top of the printable area. If the paper size is modified, this dimension's default value is scaled accordingly.

bottom-margin

The margin between the bottom of the printable area and the bottom of the page. If the paper size is modified, this dimension's default value is scaled accordingly.

ragged-bottom

If this is set to true, systems will be set at their natural spacing, neither compressed nor stretched vertically to fit the page.

ragged-last-bottom

If this is set to false, then the last page, and the last page in each section created with a `\bookpart` block, will be vertically justified in the same way as the earlier pages.

Vegeu també

Notation Reference: [Automatic scaling to paper size], pàgina 523.

Installed Files: `ly/paper-defaults-init.ly`.

Snippets: Secció “Spacing” in *Fragments de codi*.

Advertiments i problemes coneguts

The titles (from the `\header` block) are treated as a system, so **ragged-bottom** and **ragged-last-bottom** will add space between the titles and the first system of the score.

Explicitly defined paper-sizes will override any user-defined top or bottom margin settings.

4.1.4 Flexible vertical spacing \paper variables

In most cases, it is preferable for the vertical distances between certain items (such as margins, titles, systems, and separate scores) to be flexible, so that they stretch and compress nicely according to each situation. A number of `\paper` variables (listed below) are available to fine-tune the stretching behavior of these dimensions.

Note that the `\paper` variables discussed in this section do not control the spacing of staves within individual systems. Within-system spacing is controlled by grob properties, with settings typically entered inside a `\score` or `\layout` block, and not inside a `\paper` block. See Secció 4.4.1 [Flexible vertical spacing within systems], pàgina 541.

Structure of flexible vertical spacing alists

Each of the flexible vertical spacing `\paper` variables is an alist (association list) containing four *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. The reference point of a (title or top-level) markup is its highest point, and the reference point of a system is the vertical center of the nearest `StaffSymbol` – even if a non-staff line (such as a `Lyrics` context) is in the way. Values for **basic-distance** that are less than either **padding** or **minimum-distance** are not meaningful, since the resulting distance will never be less than either **padding** or **minimum-distance**.

- **minimum-distance** – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect. Values for **minimum-distance** that are less than **padding** are not meaningful, since the resulting distance will never be less than **padding**.
- **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). When positive, the significance of a particular dimension’s **stretchability** value lies only in its relation to the **stretchability** values of the other dimensions. For example, if one dimension has twice the **stretchability** of another, it will stretch twice as easily. Values should be non-negative and finite. The value **+inf.0** triggers a **programming_error** and is ignored, but **1.0e7** can be used for an almost infinitely stretchable spring. If unset, the default value is set to **basic-distance**. Note that the dimension’s propensity to *compress* cannot be directly set by the user and is equal to (**basic-distance** – **minimum-distance**).

If a page has a ragged bottom, the resulting distance is the largest of:

- **basic-distance**,
- **minimum-distance**, and
- **padding** plus the smallest distance necessary to eliminate collisions.

For multi-page scores with a ragged bottom on the last page, the last page uses the same spacing as the preceding page, provided there is enough space for that.

Specific methods for modifying alists are discussed in Secció 5.3.6 [Modifying alists], pàgina 599. The following example demonstrates the two ways these alists can be modified. The first declaration updates one key-value individually, and the second completely redefines the variable:

```
\paper {
  system-system-spacing.basic-distance = #8
  score-system-spacing =
    #'((basic-distance . 12)
      (minimum-distance . 6)
      (padding . 1)
      (stretchability . 12))
}
```

List of flexible vertical spacing \paper variables

The names of these variables follow the format *upper-lower-spacing*, where *upper* and *lower* are the items to be spaced. Each distance is measured between the reference points of the two items (see the description of the alist structure above). Note that in these variable names, the term ‘markup’ refers to both *title markups* (**bookTitleMarkup** or **scoreTitleMarkup**) and *top-level markups* (see Secció 3.1.5 [File structure], pàgina 468). All distances are measured in staff-spaces.

Default settings are defined in **ly/paper-defaults-init.ly**.

markup-system-spacing

the distance between a (title or top-level) markup and the system that follows it.

score-markup-spacing

the distance between the last system of a score and the (title or top-level) markup that follows it.

score-system-spacing

the distance between the last system of a score and the first system of the score that follows it, when no (title or top-level) markup exists between them.

system-system-spacing

the distance between two systems in the same score.

markup-markup-spacing

the distance between two (title or top-level) markups.

last-bottom-spacing

the distance from the last system or top-level markup on a page to the bottom of the printable area (i.e., the top of the bottom margin).

top-system-spacing

the distance from the top of the printable area (i.e., the bottom of the top margin) to the first system on a page, when there is no (title or top-level) markup between the two.

top-markup-spacing

the distance from the top of the printable area (i.e., the bottom of the top margin) to the first (title or top-level) markup on a page, when there is no system between the two.

Vegeu també

Notation Reference: Secció 4.4.1 [Flexible vertical spacing within systems], pàgina 541.

Installed Files: `ly/paper-defaults-init.ly`.

Snippets: Secció “Spacing” in *Fragments de codi*.

4.1.5 Horizontal spacing \paper variables

Nota: Some `\paper` dimensions are automatically scaled to the paper size, which may lead to unexpected behavior. See [Automatic scaling to paper size], pàgina 523.

\paper variables for widths and margins

Default values (before scaling) that are not listed here are defined in `ly/paper-defaults-init.ly`.

paper-width

The width of the page, unset by default. While `paper-width` has no effect on the automatic scaling of some horizontal dimensions, it does influence the `line-width` variable. If both `paper-width` and `line-width` are set, then `left-margin` and `right-margin` will also be updated. Also see `check-consistency`.

line-width

When specified in a `\paper` block this defines the horizontal extent available for the staff lines in un-indented systems. If left unspecified, the paper’s `line-width` is determined from $(\text{paper-width} - \text{left-margin} - \text{right-margin})$. If the paper’s `line-width` is specified, and both `left-margin` and `right-margin` are not, then the margins will be updated to center the systems on the page automatically. Also see `check-consistency`.

`line-widths` for individual scores can be specified in the scores’ `\layout` blocks. These values control the width of the lines produced on a score-by-score basis. If `line-width` is not specified for a score, it defaults to the paper’s `line-width`.

Setting a score's `line-width` has no effect on the paper margins. Staff lines, of a length determined by the score's `line-width`, are left-aligned within the paper area defined by the paper's `line-width`. If the score and paper `line-widths` are equal, the staff lines will extend exactly from the left margin to the right margin, but if the score's `line-width` is greater than the paper's `line-width` the staff lines will run over into the right margin.

`left-margin`

The margin between the left edge of the page and the start of the staff lines in unindented systems. If the paper size is modified, this dimension's default value is scaled accordingly. If `left-margin` is unset, and both `line-width` and `right-margin` are set, then `left-margin` is set to $(\text{paper-width} - \text{line-width} - \text{right-margin})$. If only `line-width` is set, then both margins are set to $((\text{paper-width} - \text{line-width}) / 2)$, and the systems are consequently centered on the page. Also see `check-consistency`.

`right-margin`

The margin between the right edge of the page and the end of the staff lines in non-ragged systems. If the paper size is modified, this dimension's default value is scaled accordingly. If `right-margin` is unset, and both `line-width` and `left-margin` are set, then `right-margin` is set to $(\text{paper-width} - \text{line-width} - \text{left-margin})$. If only `line-width` is set, then both margins are set to $((\text{paper-width} - \text{line-width}) / 2)$, and the systems are consequently centered on the page. Also see `check-consistency`.

`check-consistency`

If this is true (the default value), print a warning if `left-margin`, `line-width`, and `right-margin` do not exactly add up to `paper-width`, and replace each of these (except `paper-width`) with their default values (scaled to the paper size if necessary). If set to false, ignore any inconsistencies and allow systems to run off the edge of the page.

`ragged-right`

If set to true, systems will not fill the line width. Instead, systems end at their natural horizontal length. Default: `#t` for scores with only one system, and `#f` for scores with two or more systems. This variable can also be set in a `\layout` block.

`ragged-last`

If set to true, the last system in the score will not fill the line width. Instead the last system ends at its natural horizontal length. Default: `#f`. This variable can also be set in a `\layout` block.

Vegeu també

Notation Reference: [Automatic scaling to paper size], pàgina 523.

Installed Files: `ly/paper-defaults-init.ly`.

Advertiments i problemes coneguts

Explicitly defined paper-sizes will override any user-defined left or right margin settings.

`\paper` variables for two-sided mode

Default values (before scaling) are defined in `ly/paper-defaults-init.ly`.

two-sided

If set to true, use `inner-margin`, `outer-margin` and `binding-offset` to determine margins depending on whether the page number is odd or even. This overrides `left-margin` and `right-margin`.

inner-margin

The margin all pages have at the inner side if they are part of a book. If the paper size is modified, this dimension's default value is scaled accordingly. Works only with `two-sided` set to true.

outer-margin

The margin all pages have at the outer side if they are part of a book. If the paper size is modified, this dimension's default value is scaled accordingly. Works only with `two-sided` set to true.

binding-offset

The amount `inner-margin` is increased to make sure nothing will be hidden by the binding. If the paper size is modified, this dimension's default value is scaled accordingly. Works only with `two-sided` set to true.

Vegeu també

Notation Reference: [Automatic scaling to paper size], pàgina 523.

Installed Files: `ly/paper-defaults-init.ly`.

\paper variables for shifts and indents

Default values (before scaling) that are not listed here are defined in `ly/paper-defaults-init.ly`.

horizontal-shift

The amount that all systems (including titles and system separators) are shifted to the right. Default: `0.0\mm`.

indent

The level of indentation for the first system in a score. If the paper size is modified, this dimension's default value is scaled accordingly. The space within `line-width` available for the first system is reduced by this amount. `indent` may also be specified in `\layout` blocks to set indents on a score-by-score basis.

short-indent

The level of indentation for all systems in a score besides the first system. If the paper size is modified, this dimension's default value is scaled accordingly. The space within `line-width` available for systems other than the first is reduced by this amount. `short-indent` may also be specified in `\layout` blocks to set short indents on a score-by-score basis.

Vegeu també

Notation Reference: [Automatic scaling to paper size], pàgina 523.

Installed Files: `ly/paper-defaults-init.ly`.

Snippets: Secció “Spacing” in *Fragments de codi*.

4.1.6 Other \paper variables

`\paper variables for line breaking`

`max-systems-per-page`

The maximum number of systems that will be placed on a page. This is currently supported only by the `ly:optimal-breaking` algorithm. Default: unset.

`min-systems-per-page`

The minimum number of systems that will be placed on a page. This may cause pages to be overfilled if it is made too large. This is currently supported only by the `ly:optimal-breaking` algorithm. Default: unset.

`systems-per-page`

The number of systems that should be placed on each page. This is currently supported only by the `ly:optimal-breaking` algorithm. Default: unset.

`system-count`

The number of systems to be used for a score. Default: unset. This variable can also be set in a `\layout` block.

Vegeu també

Notation Reference: Secció 4.3.1 [Line breaking], pàgina 535.

`\paper variables for page breaking`

Default values not listed here are defined in `ly/paper-defaults-init.ly`

`page-breaking`

The page-breaking algorithm to use. Choices are `ly:minimal-breaking`, `ly:page-turn-breaking`, `ly:one-line-breaking`, `ly:one-line-auto-height-breaking`, and `ly:optimal-breaking` (the default).

`page-breaking-system-system-spacing`

Tricks the page breaker into thinking that `system-system-spacing` is set to something different than it really is. For example, if `page-breaking-system-system-spacing #'padding` is set to something substantially larger than `system-system-spacing #'padding`, then the page-breaker will put fewer systems on each page. Default: unset.

`page-count`

The number of pages to be used for a score, unset by default.

The following variables are effective only when `page-breaking` is set to `ly:page-turn-breaking`. Page breaks are then chosen to minimize the number of page turns. Since page turns are required on moving from an odd-numbered page to an even-numbered one, a layout in which the last page is odd-numbered will usually be favoured. Places where page turns are preferred can be indicated manually by inserting `\allowPageTurn` or automatically by including the `Page_turn_engraver` (see [Optimal page turning], pàgina 540).

If there are insufficient choices available for making suitable page turns, LilyPond may insert a blank page either within a score, between scores (if there are two or more scores), or by ending a score on an even-numbered page. The values of the following three variables may be increased to make these actions less likely.

The values are penalties, i.e., the higher the value the less likely will be the associated action relative to other choices.

`blank-page-penalty`

The penalty for having a blank page in the middle of a score. If `blank-page-penalty` is large and `ly:page-turn-breaking` is selected, then LilyPond will be less likely

to insert a page in the middle of a score. Instead, it will space out the music further to fill the blank page and the following one. Default: 5.

blank-last-page-penalty

The penalty for ending the score on an even-numbered page. If **blank-last-page-penalty** is large and **ly:page-turn-breaking** is selected, then LilyPond will be less likely to produce a score in which the last page is even-numbered. Instead, it will adjust the spacing in order to use one page more or one page less. Default: 0.

blank-after-score-page-penalty

The penalty for having a blank page after the end of one score and before the next. By default, this is smaller than **blank-page-penalty**, so that blank pages after scores are inserted in preference to blank pages within a score. Default: 2.

Vegeu també

Notation Reference: Secció 4.3.2 [Page breaking], pàgina 538, [Optimal page breaking], pàgina 539, [Optimal page turning], pàgina 540, [Minimal page breaking], pàgina 539, [One-line page breaking], pàgina 540.

Installed Files: `ly/paper-defaults-init.ly`.

\paper variables for page numbering

Default values not listed here are defined in `ly/paper-defaults-init.ly`

auto-first-page-number

The page breaking algorithm is affected by the first page number being odd or even. If set to true, the page breaking algorithm will decide whether to start with an odd or even number. This will result in the first page number remaining as is or being increased by one. Default: `#f`.

first-page-number

The value of the page number on the first page.

print-first-page-number

If set to true, a page number is printed on the first page.

print-page-number

If set to false, page numbers are not printed.

page-number-type

The type of numerals used for page numbers. Choices include `roman-lower`, `roman-upper` and `arabic`. Default: `'arabic`.

Vegeu també

Installed Files: `ly/paper-defaults-init.ly`.

Advertiments i problemes coneguts

Odd page numbers are always on the right. If you want the music to start on page 1 there must be a blank page on the back of the cover page so that page 1 is on the right hand side.

Miscellaneous \paper variables

page-spacing-weight

The relative importance of page (vertical) spacing and line (horizontal) spacing. High values will make page spacing more important. Default: 10.

print-all-headers

If set to true, this will print all headers for each `\score` in the output. Normally only the `piece` and `opus` header variables are printed. Default: `#f`.

system-separator-markup

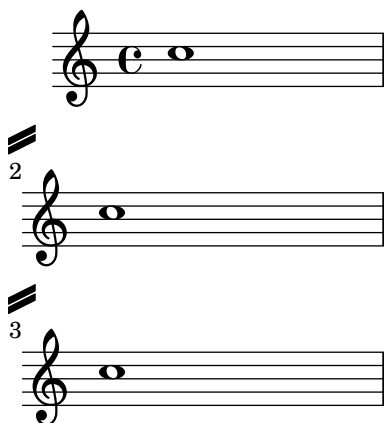
A markup object that is inserted between systems, often used for orchestral scores. Default: unset. The `\slashSeparator` markup, defined in `ly/titling-init.ly`, is provided as a sensible default, for example:

```

#(set-default-paper-size "a8")

\book {
  \paper {
    system-separator-markup = \slashSeparator
  }
  \header {
    tagline = ##f
  }
  \score {
    \relative { c''1 \break c1 \break c1 }
  }
}

```

**Vegeu també**

Installed Files: `ly/titling-init.ly`.

Snippets: Secció “Spacing” in *Fragments de codi*.

Advertiments i problemes coneguts

The default page header puts the page number and the `instrument` field from the `\header` block on a line.

4.2 Score layout

This section discusses score layout options for the `\layout` block.

4.2.1 The `\layout` block

While the `\paper` block contains settings that relate to the page formatting of the whole document, the `\layout` block contains settings for score-specific layout. To set score layout options globally, enter them in a toplevel `\layout` block. To set layout options for an individual score,

enter them in a `\layout` block inside the `\score` block, after the music. Settings that can appear in a `\layout` block include:

- the `layout-set-staff-size` scheme function,
- context modifications in `\context` blocks, and
- `\paper` variables that affect score layout.

The `layout-set-staff-size` function is discussed in the next section, Secció 4.2.2 [Setting the staff size], pàgina 533. Context modifications are discussed in a separate chapter; see Secció 5.1.4 [Modifying context plug-ins], pàgina 579, and Secció 5.1.5 [Changing context default settings], pàgina 581.

The `\paper` variables that can appear in a `\layout` block, with default values taken from the `\paper` block are:

- `line-width`, `ragged-right` and `ragged-last` (see [paper variables for widths and margins], pàgina 526)
- `indent` and `short-indent` (see [paper variables for shifts and indents], pàgina 528)
- `system-count` (see [paper variables for line breaking], pàgina 529)

Here is an example `\layout` block:

```
\layout {
  indent = 2\cm
  \context {
    \StaffGroup
    \override StaffGroup.staff-staff-spacing.basic-distance = #8
  }
  \context {
    \Voice
    \override TextScript.padding = #1
    \override Glissando.thickness = #3
  }
}
```

Multiple `\layout` blocks can be entered as toplevel expressions. This can, for example, be useful if different settings are stored in separate files and included optionally. Internally, when a `\layout` block is evaluated, a copy of the current `\layout` configuration is made, then any changes defined within the block are applied and the result is saved as the new current configuration. From the user's perspective the `\layout` blocks are combined, but in conflicting situations (when the same property is changed in different blocks) the later definitions take precedence.

For example, if this block:

```
\layout {
  \context {
    \Voice
    \override TextScript.color = #magenta
    \override Glissando.thickness = #1.5
  }
}
```

is placed after the one from the preceding example the 'padding and 'color overrides for `TextScript` are combined, but the later 'thickness override for `Glissando` replaces (or hides) the earlier one.

`\layout` blocks may be assigned to variables for reuse later, but the way this works is slightly but significantly different from writing them literally.

If a variable is defined like this:

```
layoutVariable = \layout {
  \context {
    \Voice
    \override NoteHead.font-size = #4
  }
}
```

it will hold the current `\layout` configuration with the `NoteHead.font-size` override added, but this combination is *not* saved as the new current configuration. Be aware that the ‘current configuration’ is read when the variable is defined and not when it is used, so the content of the variable is dependent on its position in the source.

The variable can then be used inside another `\layout` block, for example:

```
\layout {
  \layoutVariable
  \context {
    \Voice
    \override NoteHead.color = #red
  }
}
```

A `\layout` block containing a variable, as in the example above, does *not* copy the current configuration but instead uses the content of `\layoutVariable` as the base configuration for the further additions. This means that any changes defined between the definition and the use of the variable are lost.

If `layoutVariable` is defined (or `\included`) immediately before being used, its content is just the current configuration plus the overrides defined within it. So in the example above showing the use of `\layoutVariable` the final `\layout` block would consist of:

```
TextScript.padding = #1
TextScript.color = #magenta
Glissando.thickness = #1.5
NoteHead.font-size = #4
NoteHead.color = #red
```

plus the `indent` and the `StaffGrouper` overrides.

But if the variable had already been defined before the first `\layout` block the current configuration would now contain only

```
NoteHead.font-size = #4 % (written in the variable definition)
NoteHead.color = #red % (added after the use of the variable)
```

If carefully planned, `\layout` variables can be a valuable tool to structure the layout design of sources, and also to reset the `\layout` configuration to a known state.

Vegeu també

Notation Reference: Secció 5.1.5 [Changing context default settings], pàgina 581.

Snippets: Secció “Spacing” in *Fragments de codi*.

4.2.2 Setting the staff size

The default **staff size** is 20 points, which corresponds to a staff height of 7.03mm (one point is equal to 100/7227 of an inch, or 2540/7227 mm). The staff size may be changed in three ways:

1. To set the staff size globally for all scores in a file (or in a `\book` block, to be precise), use `set-global-staff-size`:

```
 #(set-global-staff-size 14)
```

The above example sets the global default staff size to 14pt (4.92mm) and scales all fonts accordingly.

2. To set the staff size for a single score within a book, use `layout-set-staff-size` inside that score's `\layout` block:

```
\score {
  ...
  \layout {
    #(layout-set-staff-size 14)
  }
}
```

3. To set the staff size for a single staff within a system, use the `\magnifyStaff` command. For example, traditionally engraved chamber music scores with piano often used 7mm piano staves while the other staves were typically between 3/5 and 5/7 as large (between 60% and 71%). To achieve the 5/7 proportion, use:

```
\score {
  <<
    \new Staff \with {
      \magnifyStaff #5/7
    } { ... }
    \new PianoStaff { ... }
  >>
}
```

If you happen to know which `fontSize` you wish to use, you could use the following form:

```
\score {
  <<
    \new Staff \with {
      \magnifyStaff #(magstep -3)
    } { ... }
    \new PianoStaff { ... }
  >>
}
```

To emulate the look of traditional engraving, it is best to avoid reducing the thickness of the staff lines.

Automatic font weight at different sizes

The Feta font provides musical symbols at eight different sizes. Each font is tuned for a different staff size: at a smaller size the font becomes heavier, to match the relatively heavier staff lines. The recommended font sizes are listed in the following table:

font name	staff height (pt)	staff height (mm)	use
feta11	11.22	3.9	pocket scores
feta13	12.60	4.4	
feta14	14.14	5.0	
feta16	15.87	5.6	
feta18	17.82	6.3	song books
feta20	20	7.0	standard parts
feta23	22.45	7.9	
feta26	25.2	8.9	

Vegeu també

Notation Reference: [Selecting notation font size], pàgina 215.

Snippets: Secció “Spacing” in *Fragments de codi*.

Advertiments i problemes coneguts

`layout-set-staff-size` does not change the distance between the staff lines.

4.3 Breaks

4.3.1 Line breaking

Line breaks are normally determined automatically. They are chosen so that lines look neither cramped nor loose, and consecutive lines have similar density.

To manually force a line break at a bar line, use the `\break` command:

```
\relative c'' {
  c4 c c c | \break
  c4 c c c |
}
```



By default, a `\break` command inserted in the ‘middle’ of a measure will be ignored (and a warning message will be output during the compilation of the LilyPond file). Adding an invisible bar line – `\bar ""` – before the `\break` command will force the issue:

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



A `\break` command that occurs at a bar line will also be ignored if the previous measure ends in the middle of a note (e.g., when a tuplet begins in one measure and ends in another). In this case remove the `Forbid_line_break_engraver` from the `Voice` context and, use a simultaneous music construction inserting the `\break` at the appropriate place in the second ‘voice’:

```
\new Voice \with {
  \remove "Forbid_line_break_engraver"
} \relative {
```

```
<<
{ c''2. \tuplet 3/2 { c4 c c } c2. | }
{ s1 | \break s1 | }
>>
}
```



Similarly, by default, line breaks are ignored when beams cross bar lines. Use the `\override Beam.breakable = ##t` command to force this:

```
\relative c'' {
  \override Beam.breakable = ##t
  c2. c8[ c | \break
  c8 c] c2. |
}
```



The `\noBreak` command will prevent a line break at the bar line where it is inserted.

Within a score, automatic line breaking is prevented within music lying between `\autoLineBreaksOff` and `\autoLineBreaksOn` commands. If automatic page breaks should also be prevented, the commands `\autoBreaksOff` and `\autoBreaksOn` should be used. Manual breaks are unaffected by these commands. Note that inhibiting automatic line breaks may cause music to run over the right margin if it cannot all be contained within one line.

Automatic line breaks (but not page breaks) may be enabled at single bar lines by using `\once \autoLineBreaksOn` at a bar line. This identifies a permitted rather than a forced line break.

The most basic settings influencing line spacing are `indent` and `line-width`. They are set in the `\layout` block. They control the indentation of the first line of music, and the lengths of the lines.

If `ragged-right` is set to true in the `\layout` block, then systems end at their natural horizontal length, instead of being spread horizontally to fill the whole line. This is useful for short fragments, and for checking how tight the natural spacing is.

The option `ragged-last` is similar to `ragged-right`, but affects only the last line of the piece.

```
\layout {
  indent = 0\mm
```

```

line-width = 150\mm
ragged-last = ##t
}

```

For line breaks at regular intervals use `\break` separated by skips and repeated with `\repeat`. For example, this would cause the following 28 measures (assuming 4/4 time) to be broken every 4 measures, and only there:

```

<<
\repeat unfold 7 {
  s1 \noBreak s1 \noBreak
  s1 \noBreak s1 \break
}
{ the actual music... }
>>

```

Instruccions predefinides

`\break`, `\noBreak`, `\autoBreaksOff`, `\autoBreaksOn`, `\autoLineBreaksOff`, `\autoLineBreaksOn`.

Fragments de codi seleccionats

Using an extra voice for breaks

Often it is easier to manage line and page-breaking information by keeping it separate from the music by introducing an extra voice containing only skips along with the `\break`, `pageBreak` and other layout information.

This pattern becomes especially helpful when overriding `line-break-system-details` and the other useful but long properties of `NonMusicalPaperColumnGrob`.

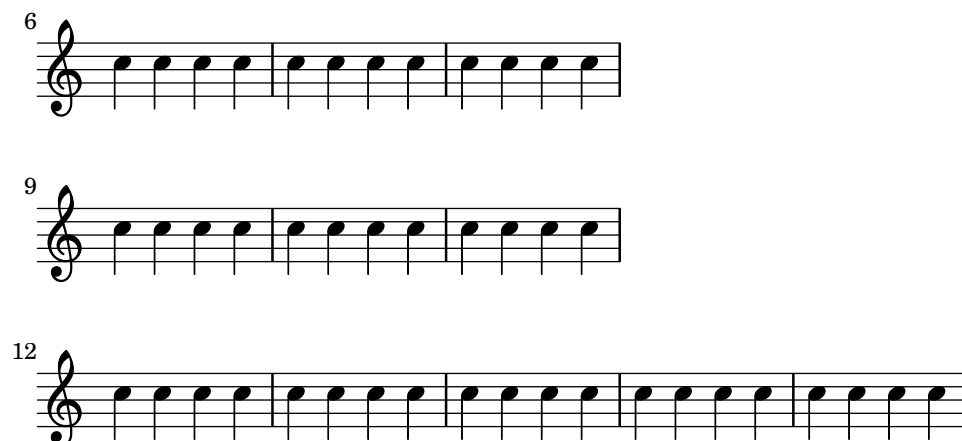
```
music = \relative c'' { c4 c c c }
```

```

\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 }
    }
  >>
}

```





Vegeu també

Notation Reference: [paper variables for line breaking], pàgina 529, Secció 4.2.1 [The layout block], pàgina 531.

Snippets: Secció “Spacing” in *Fragments de codi*.

Internals Reference: Secció “LineBreakEvent” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

Placing `\autoLineBreaksOff` or `\autoBreaksOff` before any music will cause error messages to appear. Always place these commands after some music.

4.3.2 Page breaking

This section describes the different page breaking methods, and how to modify them.

Manual page breaking

The default page breaking may be overridden by inserting `\pageBreak` or `\noPageBreak` commands. These commands are analogous to `\break` and `\noBreak`. They should be inserted at a bar line. These commands force and forbid a page-break from happening at that bar line. Of course, the `\pageBreak` command also forces a line break.

The `\pageBreak` and `\noPageBreak` commands may also be inserted at top-level, between scores and top-level markups.

Within a score, automatic page breaks are prevented within music lying between `\autoPageBreaksOff` and `\autoPageBreaksOn` commands. Manual page breaks are unaffected by these commands.

There are also analogous settings to `ragged-right` and `ragged-last` which have the same effect on vertical spacing. If `ragged-bottom` is set to `#t` the systems will not be justified vertically. When `ragged-last-bottom` is set to `#t`, as it is by default, empty space is allowed at the bottom of the final page (or the final page in each `\bookpart`). See Secció 4.1.3 [Fixed vertical spacing `\paper` variables], pàgina 523.

Page breaks are computed by the `page-breaking` function. LilyPond provides several algorithms for computing page breaks, including `ly:optimal-breaking`, `ly:page-turn-breaking` and `ly:minimal-breaking`. The default is `ly:optimal-breaking`, but the value can be changed in the `\paper` block:

```
\paper {
  page-breaking = #ly:page-turn-breaking
}
```

When a book has many scores and pages, the page breaking problem may be difficult to solve, requiring large processing time and memory. To ease the page breaking process, `\bookpart`

blocks are used to divide the book into several parts: the page breaking occurs separately on each part. Different page breaking functions may also be used in different book parts.

```
\bookpart {
  \header {
    subtitle = "Preface"
  }
  \paper {
    %% In a part consisting mostly of text,
    %% ly:minimal-breaking may be preferred
    page-breaking = #ly:minimal-breaking
  }
  \markup { ... }
  ...
}
\bookpart {
  %% In this part, consisting of music, the default optimal
  %% page breaking function is used.
  \header {
    subtitle = "First movement"
  }
  \score { ... }
  ...
}
```

Instruccions predefinides

`\pageBreak`, `\noPageBreak`, `\autoPageBreaksOn`, `\autoPageBreaksOff`.

Vegeu també

Notation Reference: [paper variables for page breaking], pàgina 529.

Snippets: Secció “Spacing” in *Fragments de codi*.

Advertiments i problemes coneguts

The `\once` prefix is ineffective with `\autoPageBreaksOn` and `\autoPageBreaksOff`. If auto page breaking is off and is then turned on to permit a page break, it must remain on for a few bars (the precise number of bars depends on the score) before being turned off, else the opportunity to break the page will not be taken.

Optimal page breaking

The `ly:optimal-breaking` function is LilyPond’s default method of determining page breaks. It attempts to find a page breaking that minimizes cramping and stretching, both horizontally and vertically. Unlike `ly:page-turn-breaking`, it has no concept of page turns.

Vegeu també

Snippets: Secció “Spacing” in *Fragments de codi*.

Minimal page breaking

The `ly:minimal-breaking` function performs minimal computations to calculate the page breaking: it fills a page with as many systems as possible before moving to the next one. Thus, it may be preferred for scores with many pages, where the other page breaking functions could be too slow or memory demanding, or a lot of texts. It is enabled using:

```
\paper {
```

```

    page-breaking = #ly:minimal-breaking
}

```

Vegeu també

Snippets: Secció “Spacing” in *Fragments de codi*.

One-line page breaking

The `ly:one-line-breaking` function is a special-purpose page breaking algorithm that puts each score on its own page, and on a single line. No titles or margins are typeset; only the score is displayed.

The page width is adjusted so that the longest score fits on one line. In particular, `paper-width`, `line-width` and `indent` variables in the `\paper` block are ignored, although `left-margin` and `right-margin` are still honored. The height of the page is left unmodified.

One-line-auto-height page breaking

The `ly:one-line-auto-height-breaking` function works just like `ly:one-line-breaking` except the page height is automatically modified to fit the height of the music. Specifically, the `paper-height` variable in the `\paper` block is set so that it spans the height of the tallest score plus the `top-margin` and `bottom-margin`.

Note that the `top-system-spacing` setting will affect the vertical position of the music. Set it to `##f` in a paper block to simply place the music between the top and bottom margins.

Optimal page turning

Often it is necessary to find a page breaking configuration so that there is a rest at the end of every second page. This way, the musician can turn the page without having to miss notes. The `ly:page-turn-breaking` function attempts to find a page breaking minimizing cramping and stretching, but with the additional restriction that it is only allowed to introduce page turns in specified places.

There are two steps to using this page breaking function. First, you must enable it in the `\paper` block, as explained in Secció 4.3.2 [Page breaking], pàgina 538. Then you must tell the function where you would like to allow page breaks.

There are two ways to achieve the second step. First, you can specify each potential page turn manually, by inserting `\allowPageTurn` into your input file at the appropriate places.

If this is too tedious, you can add a `Page_turn_engraver` to a Staff or Voice context. The `Page_turn_engraver` will scan the context for sections without notes (note that it does not scan for rests; it scans for the absence of notes. This is so that single-staff polyphony with rests in one of the parts does not throw off the `Page_turn_engraver`). When it finds a sufficiently long section without notes, the `Page_turn_engraver` will insert an `\allowPageTurn` at the final bar line in that section, unless there is a ‘special’ bar line (such as a double bar), in which case the `\allowPageTurn` will be inserted at the final ‘special’ bar line in the section.

The `Page_turn_engraver` reads the context property `minimumPageTurnLength` to determine how long a note-free section must be before a page turn is considered. The default value for `minimumPageTurnLength` is `(ly:make-moment 1/1)`. If you want to disable page turns, set it to something ‘very large’.

```

\new Staff \with { \consists "Page_turn_engraver" }
{
  a4 b c d |
  R1 | % a page turn will be allowed here
  a4 b c d |
  \set Staff.minimumPageTurnLength = #(ly:make-moment 5/2)
}

```

```

R1 | % a page turn will not be allowed here
a4 b r2 |
R1*2 | % a page turn will be allowed here
a1
}

```

When using volta repeats, the `Page_turn_engraver` will only allow a page turn during the repeat if there is enough time at the beginning and end of the repeat to turn the page back. If the repeat is too short then the `Page_turn_engraver` can be used to *disable* page turns by setting an appropriate value for the context property `minimumRepeatLengthForPageTurn`. In this case the `Page_turn_engraver` will only allow turns in repeats whose duration is longer than the value specified.

The page turning commands, `\pageTurn`, `\noPageTurn` and `\allowPageTurn`, may also be used at top-level, in top-level markups and between scores.

Instruccions predefinides

`\pageTurn`, `\noPageTurn`, `\allowPageTurn`.

Vegeu també

Notation Reference: [paper variables for line breaking], pàgina 529.

Snippets: Secció “Spacing” in *Fragments de codi*.

Advertiments i problemes coneguts

Use only one `Page_turn_engraver` per score. If there are more, they will interfere with each other.

Vegeu també

Notation Reference: Secció 4.4 [Vertical spacing], pàgina 541.

Snippets: Secció “Spacing” in *Fragments de codi*.

4.4 Vertical spacing

Vertical spacing is controlled by three things: the amount of space available (i.e., paper size and margins), the amount of space between systems, and the amount of space between staves inside a system.

4.4.1 Flexible vertical spacing within systems

Three separate mechanisms control the flexible vertical spacing within systems, one for each of the following categories:

- *ungrouped staves*,
- *grouped staves* (staves within a staff-group such as `ChoirStaff`, etc.), and
- *non-staff lines* (such as `Lyrics`, `ChordNames`, etc.).

The height of each system is determined in two steps. First, all of the staves are spaced according to the amount of space available. Then, the non-staff lines are distributed between the staves.

Note that the spacing mechanisms discussed in this section only control the vertical spacing of staves and non-staff lines within individual systems. The vertical spacing between separate systems, scores, markups, and margins is controlled by `\paper` variables, which are discussed in Secció 4.1.4 [Flexible vertical spacing `\paper` variables], pàgina 524.

Within-system spacing properties

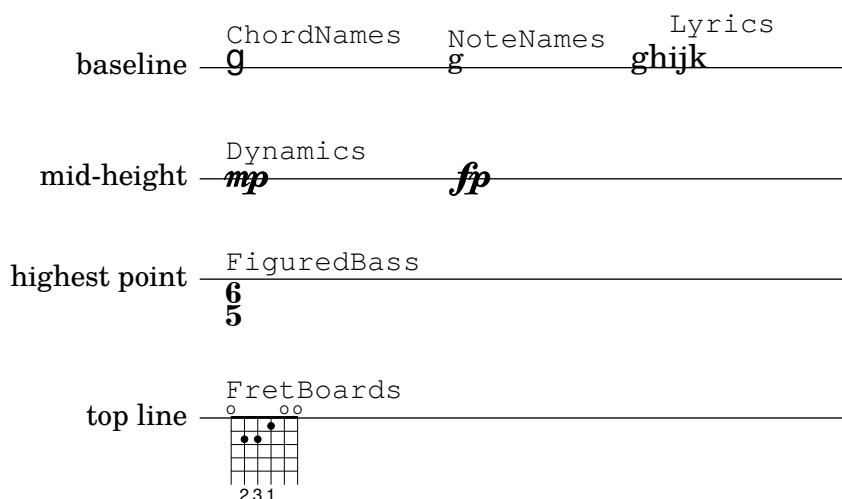
The within-system vertical spacing mechanisms are controlled by two sets of grob properties. The first set is associated with the `VerticalAxisGroup` grob, which is created by all staves and non-staff lines. The second set is associated with the `StaffGrouper` grob, which can be created by staff-groups, but only if explicitly called. These properties are described individually at the end of this section.

The names of these properties (except for `staff-affinity`) follow the format `item1-item2-spacing`, where `item1` and `item2` are the items to be spaced. Note that `item2` is not necessarily below `item1`; for example, `nonstaff-relatedstaff-spacing` will measure upwards from the non-staff line if `staff-affinity` is UP.

Each distance is measured between the *reference points* of the two items. The reference point for a staff is the vertical center of its `StaffSymbol` (i.e., the middle line if `line-count` is odd; the middle space if `line-count` is even). The reference points for individual non-staff lines are given in the following table:

Non-staff line	Reference point
ChordNames	baseline
NoteNames	baseline
Lyrics	baseline
Dynamics	mid-height of 'm'
FiguredBass	highest point
FretBoards	top line

In the following image, horizontal lines indicate the positions of these reference points:



Each of the vertical spacing grob properties (except `staff-affinity`) uses the same alist structure as the `\paper` spacing variables discussed in Secció 4.1.4 [Flexible vertical spacing `\paper` variables], pàgina 524. Specific methods for modifying alists are discussed in Secció 5.3.6 [Modifying alists], pàgina 599. Grob properties should be adjusted with an `\override` inside a `\score` or `\layout` block, and not inside a `\paper` block.

The following example demonstrates the two ways these alists can be modified. The first declaration updates one key-value individually, and the second completely re-defines the property:

```
\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))
} { ... }
```

To change any spacing settings globally, put them in the `\layout` block:

```
\layout {
  \context {
    \Staff
    \override VerticalAxisGroup.default-staff-staff-spacing.basic-distance = #10
  }
}
```

Standard settings for the vertical spacing grob properties are listed in Secció “VerticalAxisGroup” in *Referència de funcionament intern* and Secció “StaffGrouper” in *Referència de funcionament intern*. Default overrides for specific types of non-staff lines are listed in the relevant context descriptions in Secció “Contexts” in *Referència de funcionament intern*.

Properties of the VerticalAxisGroup grob

VerticalAxisGroup properties are typically adjusted with an `\override` at the `Staff` level (or equivalent).

staff-staff-spacing

Used to determine the distance between the current staff and the staff just below it in the same system, even if one or more non-staff lines (such as `Lyrics`) are placed between the two staves. Does not apply to the bottom staff of a system.

Initially, the `staff-staff-spacing` of a `VerticalAxisGroup` is a Scheme function that applies the properties of the `StaffGrouper` if the staff is part of a group, or the `default-staff-staff-spacing` of the staff otherwise. This allows staves to be spaced differently when they are grouped. For uniform spacing regardless of grouping, this function may be replaced by a flexible-spacing alist, using the complete-redefinition form of override shown above.

default-staff-staff-spacing

A flexible-spacing alist defining the `staff-staff-spacing` used for ungrouped staves, unless `staff-staff-spacing` has been explicitly set with an `\override`.

staff-affinity

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. Adjacent non-staff lines should have non-increasing `staff-affinity` from top to bottom, e.g., a non-staff line set to `UP` should not immediately follow one that is set to `DOWN`. Non-staff lines at the top of a system should use `DOWN`; those at the bottom should use `UP`. 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. Setting `staff-affinity` to `UP`, `CENTER`, or `DOWN` causes a staff to be spaced as a non-staff line.

nonstaff-relatedstaff-spacing

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. This means that the placement of a non-staff line depends on both the surrounding staves and the surrounding non-staff lines. Setting the `stretchability` of one of these types of spacing to a small value will make that spacing dominate. Setting the `stretchability` to a large value will make that spacing have little effect.

`nonstaff-nonstaff-spacing`

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.

`nonstaff-unrelatedstaff-spacing`

The distance between the current non-staff line and the 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. This can be used, for example, to require a minimum amount of padding between a Lyrics line and the staff to which it does not belong.

Properties of the StaffGrouper grob

StaffGrouper properties are typically adjusted with an `\override` at the StaffGroup level (or equivalent).

`staff-staff-spacing`

The distance between consecutive staves within the current staff-group. The `staff-staff-spacing` property of an individual staff's VerticalAxisGroup grob can be overridden with different spacing settings for that staff.

`staffgroup-staff-spacing`

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 (such as Lyrics) exist between the two staves. Does not apply to the bottom staff of a system. The `staff-staff-spacing` property of an individual staff's VerticalAxisGroup grob can be overridden with different spacing settings for that staff.

Vegeu també

Notation Reference: Secció 4.1.4 [Flexible vertical spacing `\paper` variables], pàgina 524, Secció 5.3.6 [Modifying alists], pàgina 599.

Installed Files: `ly/engraver-init.ly`, `scm/define-grobs.scm`.

Internals Reference: Secció “Contexts” in *Referència de funcionament intern*, Secció “VerticalAxisGroup” in *Referència de funcionament intern*, Secció “StaffGrouper” in *Referència de funcionament intern*.

Spacing of ungrouped staves

Staves (such as Staff, DrumStaff, TabStaff, etc.) are contexts that can contain one or more voice contexts, but cannot contain any other staves.

The following properties affect the spacing of *ungrouped* staves:

- VerticalAxisGroup properties:
 - `default-staff-staff-spacing`
 - `staff-staff-spacing`

These grob properties are described individually above; see [Within-system spacing properties], pàgina 542.

Additional properties are involved for staves that are part of a staff-group; see [Spacing of grouped staves], pàgina 546.

The following example shows how the `default-staff-staff-spacing` property can affect the spacing of ungrouped staves. The same overrides applied to `staff-staff-spacing` would have the same effect, but would also apply in cases where the staves are combined in a group or groups.

```
\layout {
  \context {
    \Staff
    \override VerticalAxisGroup.default-staff-staff-spacing =
      #'((basic-distance . 8)
        (minimum-distance . 7)
        (padding . 1))
  }
}

<<
% The very low note here needs more room than 'basic-distance
% can provide, so the distance between this staff and the next
% is determined by 'padding.
\new Staff { b,2 r | }

% Here, 'basic-distance provides enough room, and there is no
% need to compress the space (towards 'minimum-distance) to make
% room for anything else on the page, so the distance between
% this staff and the next is determined by 'basic-distance.
\new Staff { \clef bass g2 r | }

% By setting 'padding to a negative value, staves can be made to
% collide. The lowest acceptable value for 'basic-distance is 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 | }
>>
```



Vegeu també

Installed Files: `scm/define-grobs.scm`.

Snippets: Secció “Spacing” in *Fragments de codi*.

Internals Reference: Secció “VerticalAxisGroup” in *Referència de funcionament intern*.

Spacing of grouped staves

In orchestral and other large scores, it is common to place staves in groups. The space between groups is typically larger than the space between staves of the same group.

Staff-groups (such as `StaffGroup`, `ChoirStaff`, etc.) are contexts that can contain one or more staves simultaneously.

The following properties affect the spacing of staves inside staff-groups:

- `VerticalAxisGroup` properties:
 - `staff-staff-spacing`
- `StaffGrouper` properties:
 - `staff-staff-spacing`
 - `staffgroup-staff-spacing`

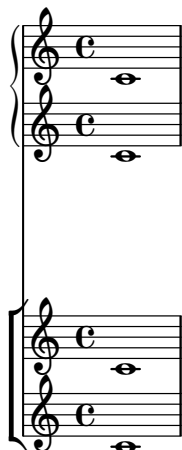
These grob properties are described individually above; see [Within-system spacing properties], pàgina 542.

The following example shows how properties of the `StaffGrouper` grob can affect the spacing of grouped staves:

```
\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 }
  >>
>>
```



Vegeu també

Installed Files: `scm/define-grobs.scm`.

Snippets: Secció “Spacing” in *Fragments de codi*.

Internals Reference: Secció “VerticalAxisGroup” in *Referència de funcionament intern*, Secció “StaffGrouper” in *Referència de funcionament intern*.

Spacing of non-staff lines

Non-staff lines (such as `Lyrics`, `ChordNames`, etc.) are contexts whose layout objects are engraved like staves (i.e., in horizontal lines within systems). Specifically, non-staff lines are non-staff contexts that contain the Secció “Axis-group-engraver” in *Referència de funcionament intern*.

The following properties affect the spacing of non-staff lines:

- `VerticalAxisGroup` properties:
 - `staff-affinity`
 - `nonstaff-relatedstaff-spacing`
 - `nonstaff-nonstaff-spacing`
 - `nonstaff-unrelatedstaff-spacing`

These grob properties are described individually above; see [Within-system spacing properties], pàgina 542.

The following example shows how the `nonstaff-nonstaff-spacing` property can affect the spacing of consecutive non-staff lines. Here, by setting the `stretchability` key to a very high value, the lyrics are able to stretch much more than usual:

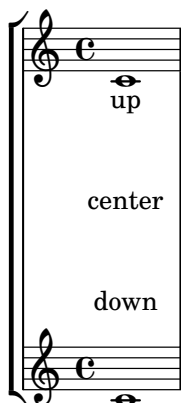
```
\layout {
  \context {
    \Lyrics
    \override VerticalAxisGroup.nonstaff-nonstaff-spacing.stretchability = #1000
  }
}

\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 }
>>

```



Vegeu també

Installed Files: `ly/engraver-init.ly`, `scm/define-grobs.scm`.

Snippets: Secció “Spacing” in *Fragments de codi*.

Internals Reference: Secció “Contexts” in *Referència de funcionament intern*, Secció “VerticalAxisGroup” in *Referència de funcionament intern*.

4.4.2 Explicit staff and system positioning

One way to understand the flexible vertical spacing mechanisms explained above is as a collection of settings that control the amount of vertical padding between staves and systems.

It is possible to approach vertical spacing in a different way using `NonMusicalPaperColumn.line-break-system-details`. While the flexible vertical spacing mechanisms specify vertical padding, `NonMusicalPaperColumn.line-break-system-details` can specify exact vertical positions on the page.

`NonMusicalPaperColumn.line-break-system-details` accepts an associative list of three different settings:

- `X-offset`
- `Y-offset`
- `alignment-distances`

Grob overrides, including the overrides for `NonMusicalPaperColumn` below, can occur in any of three different places in an input file:

- in the middle of note entry directly
- in a `\context` block
- in the `\with` block

When we override `NonMusicalPaperColumn`, we use the usual `\override` command in `\context` blocks and in the `\with` block. On the other hand, when we override `NonMusicalPaperColumn` in the middle of note entry, use the special `\overrideProperty`

command. Here are some example `NonMusicalPaperColumn` overrides with the special `\overrideProperty` command:

```
\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)))
```

To understand how each of these different settings work, we begin by looking at an example that includes no overrides at all.

```
\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' }
    }
    >>
  }
}
```



This score isolates both line-breaking and page-breaking information in a dedicated voice. This technique of creating a breaks voice will help keep layout separate from music entry as our example becomes more complicated. Also see Secció 4.3 [Breaks], pàgina 535.

By using explicit `\break` commands, the music is divided into five measures per line. Vertical spacing is from LilyPond's own defaults but the vertical startpoint of each system is set explicitly using the `Y-offset` pair in the `line-break-system-details` attribute of the `NonMusicalPaperColumn` grob:

```
\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' }
      }
    >>
  }
}
```


Note that `line-break-system-details` takes an associative list of potentially many values, but that we set only one value here. Note, too, that the `Y-offset` property here determines the exact vertical position on the page at which each new system will render.

Now that we have set the vertical startpoint of each system explicitly, we can also set the vertical distances between staves within each system manually. We do this using the `alignment-distances` subproperty of `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' }
    }
  }
}
```

```

    }
  >>
}
}

```

The image shows two systems of musical notation. Each system consists of two staves. The first system is labeled with a brace and a double greater-than sign (>>). The second system is labeled with a brace and a double less-than sign (<<). The notation consists of two staves per system, each containing a series of eighth notes.

Note that here we assign two different values to the `line-break-system-details` attribute of the `NonMusicalPaperColumn` grob. Though the `line-break-system-details` attribute alist accepts many additional spacing parameters (including, for example, a corresponding `X-offset` pair), we need only set the `Y-offset` and `alignment-distances` pairs to control the vertical startpoint of every system and every staff. Finally, note that `alignment-distances` specifies the vertical positioning of staves but not of staff groups.

```

\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 illustrating a different spacing or alignment issue. Each system consists of a single staff and a grand staff (two staves joined by a brace on the left). The first system is labeled with a '1' at the beginning. The second system is labeled with a '6' at the beginning. The third system is labeled with a '11' at the beginning. In all systems, the staves contain a sequence of eighth notes. The first system shows a single staff with a brace on the left. The second system shows a grand staff with a brace on the left. The third system shows a grand staff with a brace on the left and a single staff below it.

Some points to consider:

- When using `alignment-distances`, lyrics and other non-staff lines do not count as a staff.
- The units of the numbers passed to `X-offset`, `Y-offset` and `alignment-distances` are interpreted as multiples of the distance between adjacent staff lines. Positive values move staves and lyrics up, negative values move staves and lyrics down.
- Because the `NonMusicalPaperColumn.line-break-system-details` settings given here allow the positioning of staves and systems anywhere on the page, it is possible to violate

paper or margin boundaries or even to print staves or systems on top of one another. Reasonable values passed to these different settings will avoid this.

Vegeu també

Snippets: Secció “Spacing” in *Fragments de codi*.

4.4.3 Vertical collision avoidance

Intuitively, there are some objects in musical notation that belong to the staff and there are other objects that should be placed outside the staff. Objects belonging outside the staff include things such as rehearsal marks, text and dynamic markings (from now on, these will be called outside-staff objects). LilyPond’s rule for the vertical placement of outside-staff objects is to place them as close to the staff as possible but not so close that they collide with another object.

LilyPond uses the `outside-staff-priority` property to determine whether a grob is an outside-staff object: if `outside-staff-priority` is a number, the grob is an outside-staff object. In addition, `outside-staff-priority` tells LilyPond in which order the objects should be placed.

First, LilyPond places all the objects that do not belong outside the staff. Then it sorts the outside-staff objects according to their `outside-staff-priority` (in increasing order). One by one, LilyPond takes the outside-staff objects and places them so that they do not collide with any objects that have already been placed. That is, if two outside-staff grobs are competing for the same space, the one with the lower `outside-staff-priority` will be placed closer to the staff.

A listing of outside-staff-priorities may be found in Secció “The outside-staff-priority property” in *Manual d’aprenentatge*.

```
\relative c'' {
  c4_"Text"\pp
  r2.
  \once \override TextScript.outside-staff-priority = #1
  c4_"Text"\pp % this time the text will be closer to the staff
  r2.
  % by setting outside-staff-priority to a non-number,
  % we disable the automatic collision avoidance
  \once \override TextScript.outside-staff-priority = ##f
  \once \override DynamicLineSpanner.outside-staff-priority = ##f
  c4_"Text"\pp % now they will collide
}
```



The vertical padding around outside-staff objects can be controlled with `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-"default outside-staff-padding"
```

```

b-"default outside-staff-padding"
R1
}

```



outside-staff-padding = #3

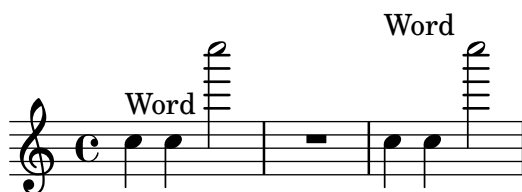
default outside-staff-padding
default outside-staff-padding

By default, outside-staff objects are placed so they avoid a horizontal collision with previously-positioned grobs. This can lead to situations in which objects are placed close to each other horizontally. As shown in the example below, setting `outside-staff-horizontal-padding` increases the horizontal spacing required, and in this case moves the text up to prevent it from getting too close to the ledger lines.

```

\relative {
  c''4^"Word" c c''2
  R1
  \once \override TextScript.outside-staff-horizontal-padding = #1
  c,,4^"Word" c c''2
}

```



Vegeu també

Snippets: Secció “Spacing” in *Fragments de codi*.

4.5 Horizontal spacing

4.5.1 Horizontal spacing overview

The spacing engine translates differences in durations into stretchable distances (‘springs’) of differing lengths. Longer durations get more space, shorter durations get less. The shortest durations get a fixed amount of space (which is controlled by `shortest-duration-space` in the Secció “SpacingSpanner” in *Referència de funcionament intern* object). The longer the duration, the more space it gets: doubling a duration adds `spacing-increment` of space to the note.

For example, the following piece contains lots of half, quarter, and 8th notes; the eighth note is followed by 1 note head width (NHW). The quarter note is followed by 2 NHW, the half by 3 NHW, etc.

```

\relative c' {
  c2 c4. c8
  c4. c8 c4. c8
  c8 c c4 c c
}

```



Normally, `spacing-increment` is set to 1.2 staff space, which is approximately the width of a note head, and `shortest-duration-space` is set to 2.0, meaning that the shortest note gets 2.4 staff space (2.0 times the `spacing-increment`) of horizontal space. This space is counted from the left edge of the symbol, so the shortest notes are generally followed by one NHW of space.

If one would follow the above procedure exactly, then adding a single 32nd note to a score that uses 8th and 16th notes, would widen up the entire score a lot. The shortest note is no longer a 16th, but a 32nd, thus adding 1 NHW to every note. To prevent this, the shortest duration for spacing is not the shortest note in the score, but rather the one which occurs most frequently.

The most common shortest duration is determined as follows: in every measure, the shortest duration is determined. The most common shortest duration is taken as the basis for the spacing, with the stipulation that this shortest duration should always be equal to or shorter than an 8th note.

These durations may also be customized. If you set the `common-shortest-duration` in Secció “SpacingSpanner” in *Referència de funcionament intern*, then this sets the base duration for spacing. The maximum duration for this base (normally an 8th), is set through `base-shortest-duration`.

Notes that are even shorter than the common shortest note are followed by a space that is proportional to their duration relative to the common shortest note. So if we were to add only a few 16th notes to the example above, they would be followed by half a NHW:

```
\relative { c''2 c4. c8 | c4. c16[ c] c4. c8 | c8 c c4 c c }
```



As explained in the *Essay on automated music engraving*, stem directions will influence spacing (see Secció “Optical spacing” in *Monografia*) and can be adjusted using the `stem-spacing-correction` property of the Secció “NoteSpacing” in *Referència de funcionament intern* object (which are generated for every Secció “Voice” in *Referència de funcionament intern* context).

The `StaffSpacing` object (generated in Secció “Staff” in *Referència de funcionament intern* context) contains the same property for controlling the stem/bar line spacing.

The following example shows this; once with the default settings and once with an exaggerated adjustment:



Proportional notation is supported; see Secció 4.5.5 [Proportional notation], pàgina 561.

Vegeu també

Essay on automated music engraving: Secció “Optical spacing” in *Monografia*.

Snippets: Secció “Spacing” in *Fragments de codi*.

Internals Reference: Secció “SpacingSpanner” in *Referència de funcionament intern*, Secció “NoteSpacing” in *Referència de funcionament intern*, Secció “StaffSpacing” in *Referència de funcionament intern*, Secció “NonMusicalPaperColumn” in *Referència de funcionament intern*.

Advertiments i problemes coneguts

There is no convenient mechanism to manually override spacing. The following work-around may be used to insert extra space into a score, adjusting the padding value as necessary.

```
\override Score.NonMusicalPaperColumn.padding = #10
```

No work-around exists for decreasing the amount of space.

4.5.2 New spacing section

New sections with different spacing parameters can be started with the `newSpacingSection` command. This is useful for sections with different notions of ‘long’ and ‘short’ notes. The `\newSpacingSection` command creates a new `SpacingSpanner` object at that musical moment.

In the following example the time signature change introduces a new section, and the 16ths notes are automatically spaced slightly wider apart.

```
\relative c' {
  \time 2/4
  c4 c8 c
  c8 c c4 c16[ c c8] c4
  \newSpacingSection
  \time 4/16
  c16[ c c8]
}
```



If the automatic spacing adjustments do not give the required spacing, manual `\overrides` may be applied to its properties. These must be applied at the same musical moment as the `\newSpacingSection` command itself and will then affect the spacing of all the following music until the properties are changed in a new spacing section, for example:

```
\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]
}
```



Vegeu també

Snippets: Secció “Spacing” in *Fragments de codi*.

Internals Reference: Secció “SpacingSpanner” in *Referència de funcionament intern*.

4.5.3 Changing horizontal spacing

Horizontal spacing may be altered with the `base-shortest-duration` property. Here we compare the same music; once without altering the property, and then altered. Larger values of `ly:make-moment` will produce smaller music. Note that `ly:make-moment` constructs a duration, so `1 4` is a longer duration than `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)
    }
  }
}
```

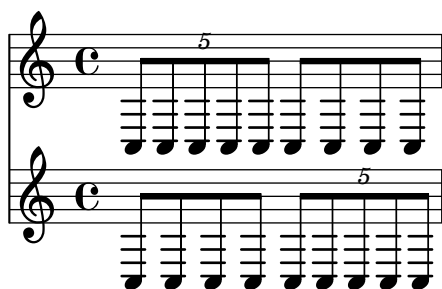




Fragments de codi seleccionats

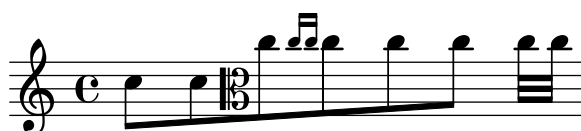
By default, spacing in tuplets depends on various non-duration factors (such as accidentals, clef changes, etc). To disregard such symbols and force uniform equal-duration spacing, use `Score.SpacingSpanner.uniform-stretching`. This property can only be changed at the beginning of a score,

```
\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
    }
  }
}
```



When `strict-note-spacing` is set, notes are spaced without regard for clefs, bar lines, and grace notes,

```
\override Score.SpacingSpanner.strict-note-spacing = ##t
\new Staff \relative {
  c''8[ c \clef alto c \grace { c16 c } c8 c c] c32[ c] }
```



Vegeu també

Snippets: Secció “Spacing” in *Fragments de codi*.

4.5.4 Line width

The most basic settings influencing the spacing are `indent` and `line-width`. They are set in the `\layout` block. They control the indentation of the first line of music, and the lengths of the lines.

If `ragged-right` is set to true in the `\layout` block, then systems ends at their natural horizontal length, instead of being spread horizontally to fill the whole line. This is useful for short fragments, and for checking how tight the natural spacing is. The normal default setting is false, but if the score has only one system the default value is true.

The option `ragged-last` is similar to `ragged-right`, but only affects the last line of the piece. No restrictions are put on that line. The result is similar to formatting text paragraphs. In a paragraph, the last line simply takes its natural horizontal length.

```
\layout {
  indent = #0
  line-width = #150
  ragged-last = ##t
}
```

Vegeu també

Snippets: Secció “Spacing” in *Fragments de codi*.

4.5.5 Proportional notation

LilyPond supports proportional notation, a type of horizontal spacing in which each note consumes an amount of horizontal space exactly equivalent to its rhythmic duration. This type of proportional spacing is comparable to horizontal spacing on top of graph paper. Some late 20th- and early 21st-century scores use proportional notation to clarify complex rhythmic relationships or to facilitate the placement of timelines or other graphics directly in the score.

LilyPond supports five different settings for proportional notation, which may be used together or alone:

- `proportionalNotationDuration`
- `uniform-stretching`
- `strict-note-spacing`
- `\remove "Separating_line_group_engraver"`
- `\override PaperColumn.used = ##t`

In the examples that follow, we explore these five different proportional notation settings and examine how these settings interact.

We start with the following one-measure example, which uses classical spacing with ragged-right turned on.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
}
```

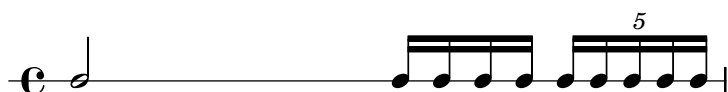


Notice that the half note which begins the measure takes up far less than half of the horizontal space of the measure. Likewise, the sixteenth notes and sixteenth-note quintuplets (or twentieth notes) which end the measure together take up far more than half the horizontal space of the measure.

In classical engraving, this spacing may be exactly what we want because we can borrow horizontal space from the half note and conserve horizontal space across the measure as a whole.

On the other hand, if we want to insert a measured timeline or other graphic above or below our score, we need proportional notation. We turn proportional notation on with the `proportionalNotationDuration` setting.

```
\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)
    }
  }
}
```



The half note at the beginning of the measure and the faster notes in the second half of the measure now occupy equal amounts of horizontal space. We could place a measured timeline or graphic above or below this example.

The `proportionalNotationDuration` setting is a context setting that lives in `Score`. Remember that context settings can appear in one of three locations within our input file – in a `\with` block, in a `\context` block, or directly in music entry preceded by the `\set` command. As with all context settings, users can pick which of the three different locations they would like to set `proportionalNotationDuration` in to.

The `proportionalNotationDuration` setting takes a single argument, which is the reference duration against that all music will be spaced. The LilyPond Scheme function `make-moment` takes two arguments – a numerator and denominator which together express some fraction of a whole note. The call `(ly:make-moment 1/20)` therefore produces a reference duration of a twentieth note. Values such as `(ly:make-moment 1/16)`, `(ly:make-moment 1/8)`, and `(ly:make-moment 3/97)` are all possible as well.

How do we select the right reference duration to pass to `proportionalNotationDuration`? Usually by a process of trial and error, beginning with a duration close to the fastest (or smallest) duration in the piece. Smaller reference durations space music loosely; larger reference durations space music tightly.

```
\score {
  <<
    \new RhythmicStaff {
      c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
    }
  >>
}
```

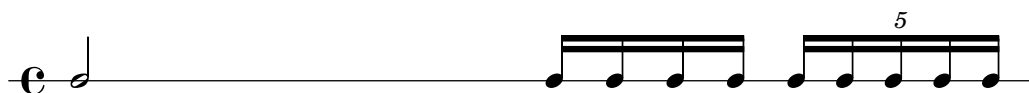
```

    }
  >>
  \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)
    }
  }
}

```



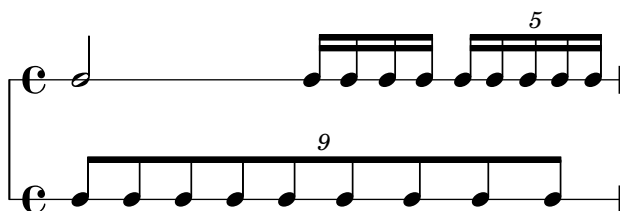
Note that too large a reference duration – such as the eighth note, above – spaces music too tightly and can cause note head collisions. Also that proportional notation in general takes up

more horizontal space than classical spacing. Proportional spacing provides rhythmic clarity at the expense of horizontal space.

Next we examine how to optimally space overlapping triplets.

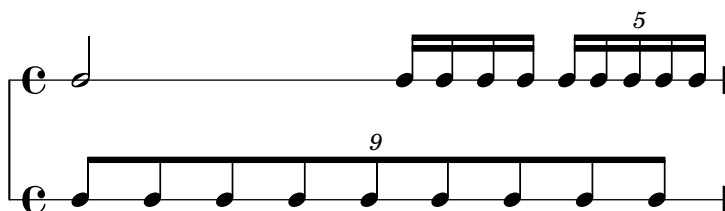
We start by examining what happens to our original example, with classical spacing, when we add a second staff with a different type of triplet.

```
\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 }
    }
  >>
}
```



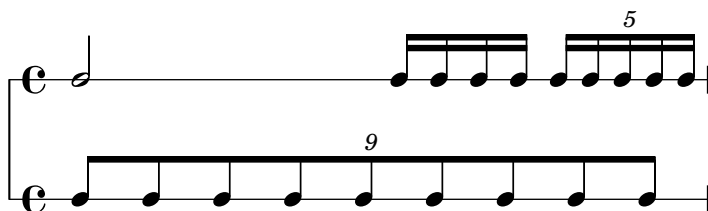
The spacing is bad because the evenly spaced notes of the bottom staff do not stretch uniformly. Classical engravings include very few complex triplets and so classical engraving rules can generate this type of result. Setting `proportionalNotationDuration` fixes this.

```
\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)
    }
  }
}
```



But if we look very carefully we can see that notes of the second half of the 9-tuplet space ever so slightly more widely than the notes of the first half of the 9-tuplet. To ensure uniform stretching, we turn on `uniform-stretching`, which is a property of `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
    }
  }
}
```



Our two-staff example now spaces exactly, our rhythmic relationships are visually clear, and we can include a measured timeline or graphic if we want.

Note that the LilyPond's proportional notation package expects that all proportional scores set the `SpacingSpanner`'s `'uniform-stretching` attribute to `##t`. Setting `proportionalNotationDuration` without also setting the `SpacingSpanner`'s `'uniform-stretching` attribute to `##t` will, for example, cause Skips to consume an incorrect amount of horizontal space.

The `SpacingSpanner` is an abstract grob that lives in the `Score` context. As with our settings of `proportionalNotationDuration`, overrides to the `SpacingSpanner` can occur in any of three different places in our input file – in the `Score` `\with block`, in a `Score` `\context block`, or in note entry directly.

There is by default only one `SpacingSpanner` per `Score`. This means that, by default, `uniform-stretching` is either turned on for the entire score or turned off for the entire score. We can, however, override this behavior and turn on different spacing features at different places in the score. We do this with the command `\newSpacingSection`. See Secció 4.5.2 [New spacing section], pàgina 558, for more info.

Next we examine the effects of the `Separating_line_group_engraver` and see why proportional scores frequently remove this engraver. The following example shows that there is a small amount of “prefatory” space just before the first note in each system.

```
\paper {
  indent = #0
}
```

```
\new Staff {
```

```

c'1
\break
c'1
}

```



The amount of this prefatory space is the same whether after a time signature, a key signature or a clef. `Separating_line_group_engraver` is responsible for this space. Removing `Separating_line_group_engraver` reduces this space to zero.

```

\paper {
  indent = #0
}

\new Staff \with {
  \remove "Separating_line_group_engraver"
} {
  c'1
  \break
  c'1
}

```



non-musical elements like time signatures, key signatures, clefs and accidentals are problematic in proportional notation. None of these elements has rhythmic duration. But all of these elements consume horizontal space. Different proportional scores approach these problems differently.

It may be possible to avoid spacing problems with key signatures simply by not having any. This is a valid option since most proportional scores are contemporary music. The same may be true of time signatures, especially for those scores that include a measured timeline or other graphic. But these scores are exceptional and most proportional scores include at least some time signatures. Clefs and accidentals are even more essential.

So what strategies exist for spacing non-musical elements in a proportional context? One good option is the `strict-note-spacing` property of `SpacingSpanner`. Compare the two scores below:

```

\new Staff {
  \set Score.proportionalNotationDuration = #(ly:make-moment 1/16)

```

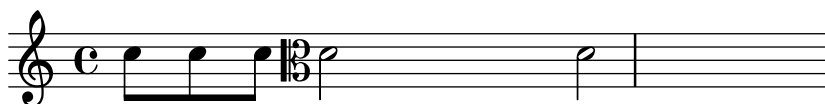
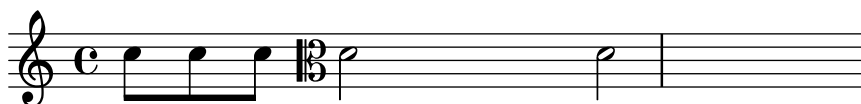


```

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
}

```



Both scores are proportional, but the spacing in the first score is too loose because of the clef change. The spacing of the second score remains strict, however, because `strict-note-spacing` is turned on. Turning on `strict-note-spacing` causes the width of time signatures, key signatures, clefs and accidentals to play no part in the spacing algorithm.

In addition to the settings given here, there are other settings that frequently appear in proportional scores. These include:

- `\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"` in the Voice context

These settings space grace notes strictly, extend tuplet brackets to mark both rhythmic start- and stop-points, and allow spanning elements to break across systems and pages. See the respective parts of the manual for these related settings.

Vegeu també

Notation Reference: Secció 4.5.2 [New spacing section], pàgina 558.

Snippets: Secció “Spacing” in *Fragments de codi*.

4.6 Fitting music onto fewer pages

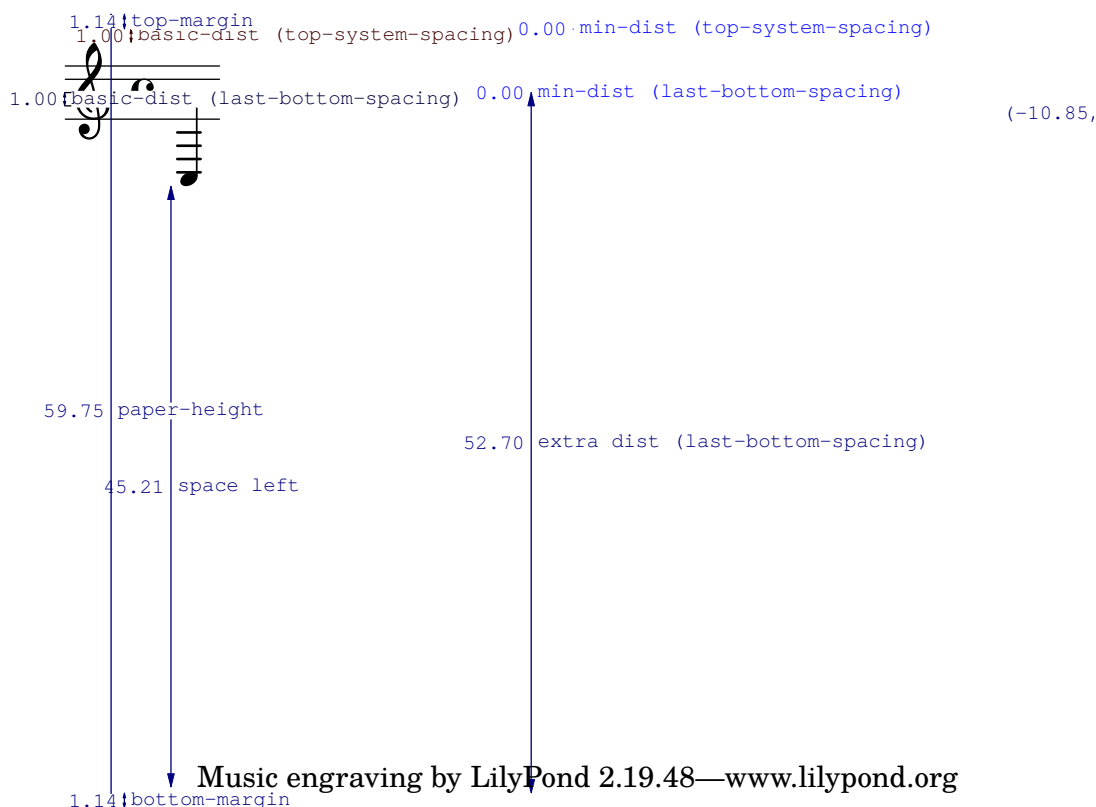
Sometimes you can end up with one or two staves on a second (or third, or fourth...) page. This is annoying, especially if you look at previous pages and it looks like there is plenty of room left on those.

When investigating layout issues, `annotate-spacing` is an invaluable tool. This command prints the values of various layout spacing variables; for more details see the following section, Secció 4.6.1 [Displaying spacing], pàgina 568.

4.6.1 Displaying spacing

To graphically display the dimensions of vertical layout variables that may be altered for page formatting, set `annotate-spacing` in the `\paper` block:

```
\book {
  \score { { c4 } }
  \paper { annotate-spacing = ##t }
}
```



All layout dimensions are displayed in staff-spaces, regardless of the units specified in the `\paper` or `\layout` block. In the above example, `paper-height` has a value of 59.75 `staff-spaces`, and the `staff-size` is 20 points (the default value). Note that:

$$\begin{aligned}
 1 \text{ point} &= (25.4/72.27) \text{ mm} \\
 1 \text{ staff-space} &= (\text{staff-size})/4 \text{ pts} \\
 &= (\text{staff-size})/4 * \\
 &= (25.4/72.27) \text{ mm}
 \end{aligned}$$

In this case, one `staff-space` is approximately equal to 1.757mm. Thus the `paper-height` measurement of 59.75 `staff-spaces` is equivalent to 105 millimeters, the height of a6 paper in landscape orientation. The pairs (a,b) are intervals, where a is the lower edge and b the upper edge of the interval.

Vegeu també

Notation Reference: Secció 4.2.2 [Setting the staff size], pàgina 533.

Snippets: Secció “Spacing” in *Fragments de codi*.

4.6.2 Changing spacing

The output of `annotate-spacing` reveals vertical dimensions in great detail. For details about modifying margins and other layout variables, see Secció 4.1 [Page layout], pàgina 521.

Other than margins, there are a few other options to save space:

- Force systems to move as close together as possible (to fit as many systems as possible onto a page) while being spaced so that there is no blank space at the bottom of the page.

```
\paper {
  system-system-spacing = #'((basic-distance . 0.1) (padding . 0))
  ragged-last-bottom = ##f
  ragged-bottom = ##f
}
```

- Force the number of systems. This can help in two ways. Just setting a value, even the same value as the number of systems being typeset by default, will sometimes cause more systems to be fitted onto each page, as an estimation step is then bypassed, giving a more accurate fit to each page. Also, forcing an actual reduction in the number of systems may save a further page. For example, if the default layout has 11 systems, the following assignment will force a layout with 10 systems.

```
\paper {
  system-count = #10
}
```

- Force the number of pages. For example, the following assignment will force a layout with 2 pages.

```
\paper {
  page-count = #2
}
```

- Avoid (or reduce) objects that increase the vertical size of a system. For example, volta brackets for alternative repeat endings require extra space. If these endings are spread over two systems, they take up more space than if they were on the same system. As another example, dynamics that ‘stick out’ of a system can be moved closer to the staff:

```
\relative e' {
  e4 c g\ff c
  e4 c g-\tweak X-offset #-2.7 \ff c
}
```



- Alter the horizontal spacing via `SpacingSpanner`. For more details, see Secció 4.5.3 [Changing horizontal spacing], pàgina 559. The following example illustrates the default spacing:

```
\score {
  \relative {
    g'4 e e2 |
    f4 d d2 |
    c4 d e f |
    g4 g g2 |
    g4 e e2 |
  }
}
```

}



The next example modifies `common-shortest-duration` from a value of `1/4` to `1/2`. The quarter note is the most common and shortest duration in this example, so by making this duration longer, a ‘squeezing’ effect occurs:

```
\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)
    }
  }
}
```



The `common-shortest-duration` property cannot be modified dynamically, so it must always be placed in a `\context` block so that it applies to the whole score.

Vegeu també

Notation Reference: Secció 4.1 [Page layout], pàgina 521, Secció 4.5.3 [Changing horizontal spacing], pàgina 559.

Snippets: Secció “Spacing” in *Fragments de codi*.

5 Changing defaults

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 Learning Manual, see Secció “Tweaking output” in *Manual d’aprenentatge*. 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: Secció “the Internals Reference” in *Referència de funcionament intern*. 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 `#`.¹

5.1 Interpretation contexts

This section describes what contexts are, and how to modify them.

Vegeu també

Learning Manual: Secció “Contexts and engravers” in *Manual d’aprenentatge*.

Installed Files: `ly/engraver-init.ly`, `ly/performer-init.ly`.

Snippets: Secció “Contexts and engravers” in *Fragments de codi*.

Internals Reference: Secció “Contexts” in *Referència de funcionament intern*, Secció “Engravers and Performers” in *Referència de funcionament intern*.

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], pàgina 581).

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 ([Automatic part combining], pàgina 178) or creating music quotes ([Quoting other voices], pàgina 206).

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

¹ Secció “Scheme tutorial” in *Extendre*, contains a short tutorial on entering numbers, lists, strings, and symbols in Scheme.

[Automatic syllable durations], pàgina 259. 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.

Vegeu també

Installed Files: `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 [Formatting cue notes], pàgina 209. 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 Secció “Organizing pieces with variables” in *Manual d’aprenentatge*.

- `\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], pàgina 259.

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], pàgina 581. 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], pàgina 584.

Vegeu també

Learning Manual: Secció “Organizing pieces with variables” in *Manual d’aprenentatge*.

Notation Reference: [Changing just one specific context], pàgina 584, [Automatic syllable durations], pàgina 259.

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.

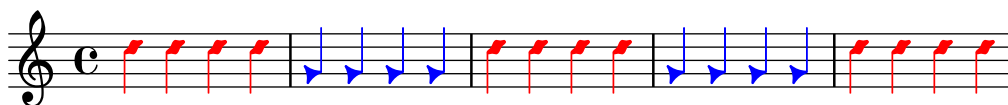
Any context can be kept alive by ensuring it has something to do at every musical moment. **Staff** contexts are kept alive by ensuring one of their voices is kept alive. One way of doing this is to add spacer rests to a voice in parallel with the real music. These need to be added to every **Voice** 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
  >>
}
```



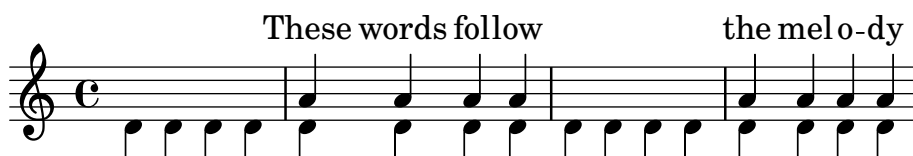
The following example shows how a sporadic melody line with lyrics might be written using this approach. In a real situation the melody and accompaniment would consist of several different sections, of course.

```
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:

```

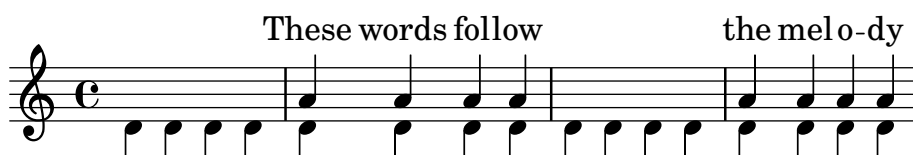
melody = \relative {
  s1 % skip a bar
  a'4 a a a
  s1 % skip a bar
  a4 a a a
}
accompaniment = \relative {
  d'4 d d d
  d4 d d d
  d4 d d d
  d4 d d d
}
words = \lyricmode { These words fol -- low the mel -- o -- dy }

```

```

\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 [Internals Reference](#) \mapsto [Translation](#) \mapsto [Engravers](#). Every context described in [Internals Reference](#) \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 Secció “Visibility and color of objects” in *Manual d’aprenentatge*.

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"
  }
}
}

```



Advertiments i problemes coneguts

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`.

Vegeu també

Installed Files: `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 Secció 5.3 [Modifying properties], pàgina 593. 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], pàgina 571).

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]
  }
}
```

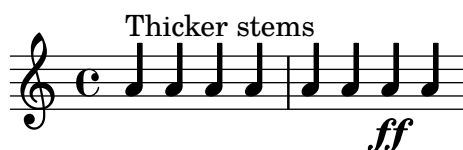
The following types of settings may be specified:

- 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
 }
}

```

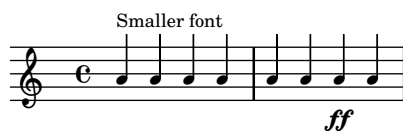


- Directly setting a context property

```

\score {
 \relative {
 a'4~"Smaller font" a a a
 a4 a a\ff a
 }
 \layout {
 \context {
 \Staff
 fontSize = #-4
 }
 }
}

```



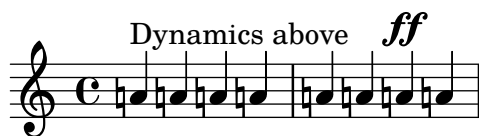
- A predefined command such as `\dynamicUp` or a music expression like `\accidentalStyle dodecaphonic`

```

\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], pàgina 584.

```
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], pàgina 573. 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:

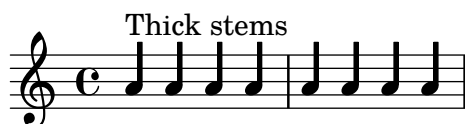
```
\new Staff \with { [context settings for this context instance only] }
{
 ...
}
```

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:

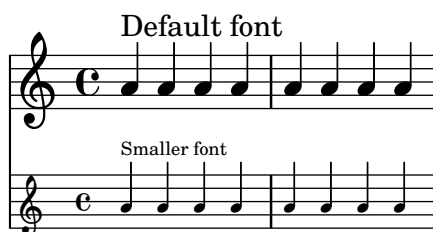
- An `\override` command, but with the context name omitted

```
\score {
 \new Staff {
 \new Voice \with { \override Stem.thickness = #4.0 }
 {
 \relative {
 a'4~"Thick stems" a a a
 a4 a a a
 }
 }
 }
}
```



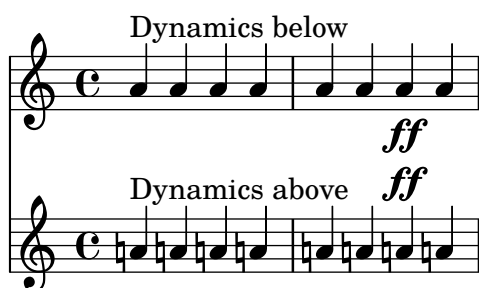
- Directly setting a context property

```
\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.

## Vegeu també

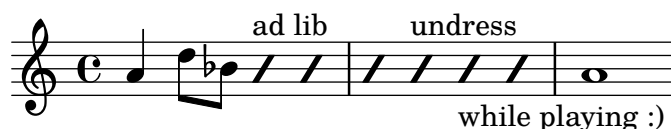
Learning Manual: Secció “Modifying context properties” in *Manual d’aprenentatge*.

Notation Reference: Secció 5.1.1 [Contexts explained], pàgina 571, [Bottom-level contexts - voices], pàgina 573, Secció 5.3.2 [The set command], pàgina 593, Secció 5.3.3 [The override command], pàgina 595, Secció 4.2.1 [The `\layout` block], pàgina 531.

### 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.

## Vegeu també

Internals Reference: Secció “`Note_heads_engraver`” in *Referència de funcionament intern*, Secció “`Text_engraver`” in *Referència de funcionament intern*, Secció “`Rhythmic_column_engraver`” in *Referència de funcionament intern*, Secció “`Pitch_squash_engraver`” in *Referència de funcionament intern*.

### 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.

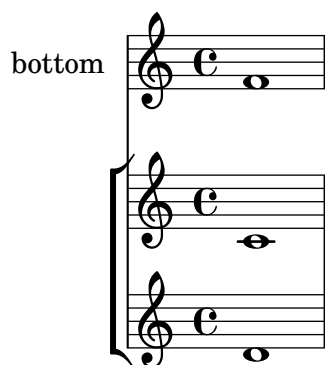
The “accepts” list of a context can be changed with the `\accepts` or `\denies` commands. `\accepts` adds a context to the “accepts” list and `\denies` removes a context from the list.

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 }
 >>
}

```

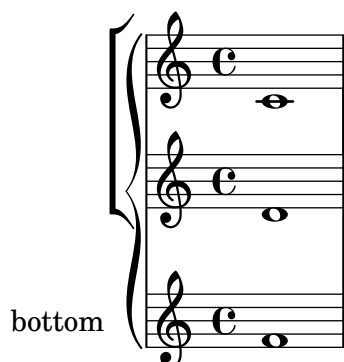


However, by using the `\accepts` command, `StaffGroup` can be added to the `GrandStaff` context:

```

\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 Secció “Nesting music expressions” in *Manual d’aprenentatge*, Secció 1.6.2 [Modifying single staves], pàgina 194, and Secció 2.1.2 [Techniques specific to lyrics], pàgina 267.

## Vegeu també

Learning Manual: Secció “Nesting music expressions” in *Manual d’aprenentatge*.

Notation Reference: Secció 1.6.2 [Modifying single staves], pàgina 194, Secció 2.1.2 [Techniques specific to lyrics], pàgina 267.

Application Usage: Secció “An extra staff appears” in *Utilització del programa*.

Installed Files: `ly/engraver-init.ly`.

## 5.2 Explaining the Internals Reference

### 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], pàgina 219), you will notice:

#### See also

Internals Reference: Secció “Fingering” in *Referència de funcionament intern*.

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 Secció “Fingering” in *Referència de funcionament intern*. At the top of the page, you will see

Fingering objects are created by: Secció “Fingering-engraver” in *Referència de funcionament intern* and Secció “New\_fingering-engraver” in *Referència de funcionament intern*.

By following related links inside the program reference, we can follow the flow of information within the program:

- Secció “Fingering” in *Referència de funcionament intern*: Secció “Fingering” in *Referència de funcionament intern* objects are created by: Secció “Fingering-engraver” in *Referència de funcionament intern*
- Secció “Fingering-engraver” in *Referència de funcionament intern*: Music types accepted: Secció “fingering-event” in *Referència de funcionament intern*
- Secció “fingering-event” in *Referència de funcionament intern*: Music event type **fingering-event** is in Music expressions named Secció “FingeringEvent” in *Referència de funcionament intern*

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 Secció “Translation” in *Referència de funcionament intern*, and the Secció “Backend” in *Referència de funcionament intern*. 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 Secció “Fingering” in *Referència de funcionament intern*. 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 Secció “grob-interface” in *Referència de funcionament intern*.

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 Secció “Fingering” in *Referència de funcionament intern* page at the bottom

This object supports the following interfaces: Secció “item-interface” in *Referència de funcionament intern*, Secció “self-alignment-interface” in *Referència de funcionament intern*, Secció “side-position-interface” in *Referència de funcionament intern*, Secció “text-interface” in *Referència de funcionament intern*, Secció “text-script-interface” in *Referència de funcionament intern*, Secció “font-interface” in *Referència de funcionament intern*, Secció “finger-interface” in *Referència de funcionament intern*, and Secció “grob-interface” in *Referència de funcionament intern*.

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 Secció “Other sources of information” in *Manual d’aprenentatge*) `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 Internals Reference 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 will insert “three staff spaces” worth of distance between the note and a fingering mark:

```
\once \override Voice.Fingering.padding = #3
```

Inserting the padding before the fingering object is created results in the following:

```
\once \override Voice.Fingering.padding = #3
```

`c''-2`



In this case, the context for this tweak is **Voice**. See Secció “Fingering-engraver” in *Referència de funcionament intern* plug-in, which says:

Fingering-engraver is part of contexts: . . . Secció “Voice” in *Referència de funcionament intern*

### 5.2.4 Naming conventions

Another thing that is needed, is an overview of the various naming conventions:

- scheme functions: lowercase-with-hyphens (also includes one-word names)
- LilyPond-specific 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 LilyPond-specific?

## 5.3 Modifying properties

### 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.

### Vegeu també

Internals Reference: Secció “Backend” in *Referència de funcionament intern*, Secció “All layout objects” in *Referència de funcionament intern*, Secció “OverrideProperty” in *Referència de funcionament intern*, Secció “RevertProperty” in *Referència de funcionament intern*, Secció “PropertySet” in *Referència de funcionament intern*.

### Advertiments i problemes coneguts

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**).

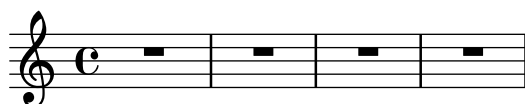
```
\set Score.autoBeaming = ##f
\relative {
 e''8 e e e
 \set autoBeaming = ##t
 e8 e e e
} \
\relative {
 c''8 c c c c8 c c c
}
```



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 internals reference, see Translation  $\mapsto$  Tunable context properties.

## Vegeu també

Internals Reference: Secció “Tunable context properties” in *Referència de funcionament intern*.

### 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
}
>>

```



## Vegeu també

Internals Reference: Secció “Backend” in *Referència de funcionament intern*

### 5.3.4 The `\tweak` command

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 Secció “Tweaking methods” in *Manual d’aprenentatge*.

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 Secció “Displaying music expressions” in *Extendre*. 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.



## Vegeu també

Learning Manual: Secció “Tweaking methods” in *Manual d’aprenentatge*.

Extending LilyPond: Secció “Displaying music expressions” in *Extendre*.

## Advertiments i problemes coneguts

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`

The `\set` and `\override` commands manipulate properties associated with contexts. In both cases, the properties follow a *hierarchy of contexts*; properties that are not set themselves in a context will still show the values of their respective parent’s context.

The lifetime and value of a context property is dynamic and only available when music is being interpreted (i.e., ‘iterated’). At the time of the context’s creation, properties are initialized from its corresponding definitions (along with any other modifications) of that context. Any subsequent changes are achieved with any ‘property-setting’ commands that are within the music itself.

Graphical Object (or “grob”) definitions are a *special* category of context properties as their structure and use is different from that of normal context properties. Unlike normal context properties, grob definitions are subdivided into *grob properties*.

Also, in contrast to normal context properties, grob definitions have their own internal ‘book-keeping’ used to keep track of their own individual grob properties and any sub-properties. This means that it is possible to define those parts within different contexts and yet still have the overall grob definition at the time of grob creation from all the pieces provided amongst the current context and its parent(s).

A grob 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) can then change (or add to) the grob’s initial properties. However, this does not affect the context’s own grob definition.

What LilyPond calls *grob properties* in the context of ‘user-level’ tweaks are really the properties of a *context’s* own grob definition.

Grob definitions are accessed with a different set of commands and are manipulated using `\override` and `\revert` and have a name starting with a capital letter (e.g., ‘`NoteHead`’); whereas normal context properties are manipulated using `\set` and `\unset` and are named starting with a lowercase letter.

The commands `\tweak` and `\overrideProperty` change grob properties by bypassing all context properties completely and, instead, catch grobs as they are being created, setting properties on them for a music event (`\tweak`) or, in the case of `\overrideProperty` for a specific override.

### 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.

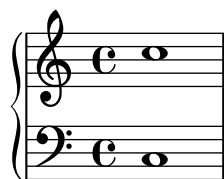
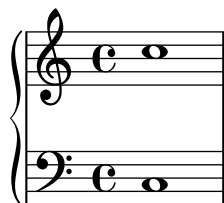
For example, to reduce the space between adjacent staves in a staff-group, use the `staff-staff-spacing` property of the `StaffGrouper` grob. The property is an alist with four keys: `basic-distance`, `minimum-distance`, `padding`, and `stretchability`. The standard settings for this property are listed in the “Backend” section of the Internals Reference (see Secció “StaffGrouper” in *Referència de funcionament intern*):

```
'((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.

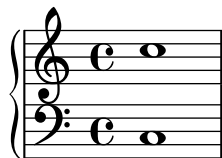
Now suppose we want the staves to be as close as possible without overlapping. The simplest way to do this is to set all four alist keys to zero. However, it is not necessary to enter four nested declarations, one for each key. Instead, the property can be completely re-defined with one declaration, 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 Useful concepts and properties

### 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 Secció 2.7 [Chord notation], pàgina 404. 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], pàgina 409.

### *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], pàgina 382.

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], pàgina 382.

### *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], pàgina 418.

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], pàgina 417.

### *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], pàgina 334.

To create fret diagrams above a staff, you have two choices. You can either use the `FretBoards` context (see [Automatic fret diagrams], pàgina 372) or you can enter them as a markup above the notes using the `\fret-diagram` command (see [Fret diagram markups], pàgina 352).

### *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 Secció 2.1 [Vocal music], pàgina 255.

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 Secció A.11 [Text markup commands], pàgina 674.

### *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 Secció “Within-staff objects” in *Manual d’aprenentatge*.

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 Secció 1.5.2 [Multiple voices], pàgina 169.

## Vegeu també

Learning Manual: Secció “Within-staff objects” in *Manual d'aprenentatge*.

Notation Reference: Secció 1.5.2 [Multiple voices], pàgina 169.

### 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 Secció 4.1 [Page layout], pàgina 521.

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 Secció 4.2.2 [Setting the staff size], pàgina 533.

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 Secció “Length and thickness of objects” in *Manual d'aprenentatge*.

## Vegeu també

Learning Manual: Secció “Length and thickness of objects” in *Manual d'aprenentatge*.

Notation Reference: Secció 4.1 [Page layout], pàgina 521, Secció 4.2.2 [Setting the staff size], pàgina 533.

### 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.

## Vegeu també

Notation Reference: Secció 5.4.3 [Distances and measurements], pàgina 604.

### 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.

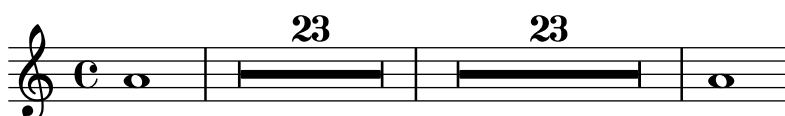
##### *The minimum-length property*

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'
```

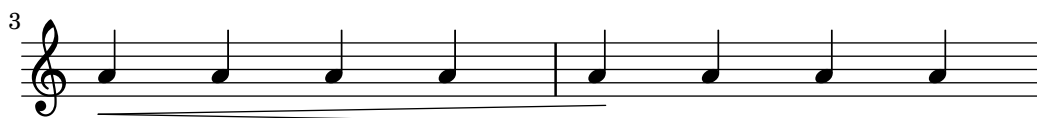
```
% effective only when both overrides are present
\once \override Beam.minimum-length = #20
\once \override Beam.springs-and-rods = #ly:spanner::set-spacing-rods
e'8 e' e' e'
```



### *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-points 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 `\!`.

## Vegeu també

Internals Reference: Secció “TextSpanner” in *Referència de funcionament intern*, Secció “Glissando” in *Referència de funcionament intern*, Secció “VoiceFollower” in *Referència de funcionament intern*, Secció “TrillSpanner” in *Referència de funcionament intern*, Secció “line-spanner-interface” in *Referència de funcionament intern*.

### 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 Learning Manual introduces these four techniques, see Secció “Visibility and color of objects” in *Manual d’aprenentatge*.

There are also a few other techniques which are specific to certain layout objects. These are covered under Special considerations.

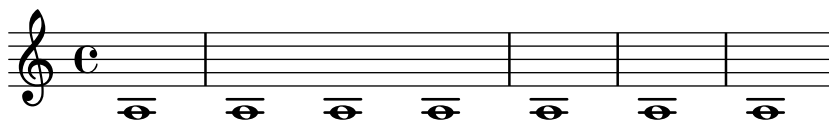
## Removing the stencil

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`:

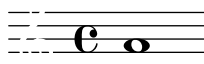
```
a'4 a'
\once \hide NoteHead
a' a'
```



## 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:

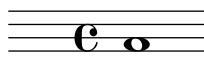
```
\override Staff.Clef.color = #white
a'1
```



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 Secció “Visibility and color of objects” in *Manual d’aprenentatge*. 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:

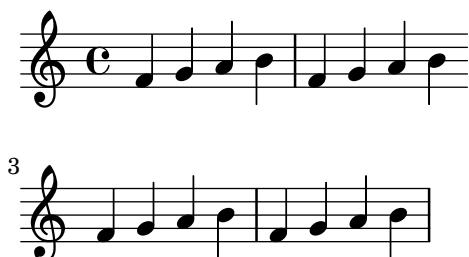
| Function form                        | Vector form              | Before break | At no break | After break |
|--------------------------------------|--------------------------|--------------|-------------|-------------|
| <code>all-visible</code>             | <code>##t ##t ##t</code> | yes          | yes         | yes         |
| <code>begin-of-line-visible</code>   | <code>##f ##f ##t</code> | no           | no          | yes         |
| <code>center-visible</code>          | <code>##f ##t ##f</code> | no           | yes         | no          |
| <code>end-of-line-visible</code>     | <code>##t ##f ##f</code> | yes          | no          | no          |
| <code>begin-of-line-invisible</code> | <code>##t ##t ##f</code> | yes          | yes         | no          |
| <code>center-invisible</code>        | <code>##t ##f ##t</code> | yes          | no          | yes         |
| <code>end-of-line-invisible</code>   | <code>##f ##t ##t</code> | no           | yes         | yes         |
| <code>all-invisible</code>           | <code>##f ##f ##f</code> | 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                      |
|----------------------------------|---------------|--------------------------------------|
| <code>BarLine</code>             | Score         | calculated                           |
| <code>BarNumber</code>           | Score         | <code>begin-of-line-visible</code>   |
| <code>BreathingSign</code>       | Voice         | <code>begin-of-line-invisible</code> |
| <code>Clef</code>                | Staff         | <code>begin-of-line-visible</code>   |
| <code>Custos</code>              | Staff         | <code>end-of-line-visible</code>     |
| <code>DoublePercentRepeat</code> | Voice         | <code>begin-of-line-invisible</code> |
| <code>KeyCancellation</code>     | Staff         | <code>begin-of-line-invisible</code> |
| <code>KeySignature</code>        | Staff         | <code>begin-of-line-visible</code>   |
| <code>ClefModifier</code>        | Staff         | <code>begin-of-line-visible</code>   |
| <code>RehearsalMark</code>       | Score         | <code>end-of-line-invisible</code>   |
| <code>TimeSignature</code>       | Staff         | <code>all-visible</code>             |

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 the 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 the next section.
- Double percent repeats are either *all printed* or *all suppressed*. Use `begin-of-line-invisible` to print them and `all-invisible` to suppress them.
- Key signature – see the next section.
- ClefModifier – see the next section.

## 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.

### Vegeu també

Learning Manual: Secció “Visibility and color of objects” in *Manual d’aprenentatge*.

## 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 Secció 5.4.6 [Spanners], pàgina 605.

### 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 Advanced tweaks

This section discusses various approaches to fine tuning the appearance of the printed score.

### Vegeu també

Learning Manual: Secció “Tweaking output” in *Manual d’aprenentatge*, Secció “Other sources of information” in *Manual d’aprenentatge*.

Notation Reference: Secció 5.2 [Explaining the Internals Reference], pàgina 590, Secció 5.3 [Modifying properties], pàgina 593.

Extending LilyPond: Secció “Interfaces for programmers” in *Extendre*.

Installed Files: `scm/define-grobs.scm`.

Snippets: Secció “Tweaks and overrides” in *Fragments de codi*.

Internals Reference: Secció “All layout objects” in *Referència de funcionament intern*.

### 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 Secció “self-alignment-interface” in *Referència de funcionament intern* and Secció “side-position-interface” in *Referència de funcionament intern*.

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.

### Vegeu també

Notation Reference: [Using the break-alignable-interface], pàgina 621.

Extending LilyPond: Secció “Callback functions” in *Extendre*.

## 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:

| side-axis<br>property | direction<br>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

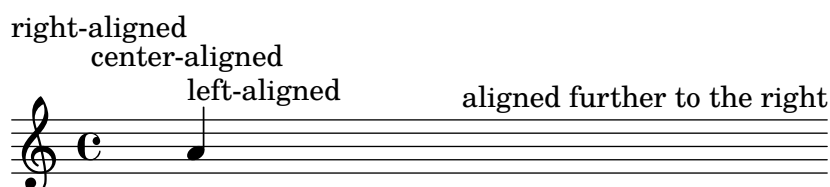
### *Self-aligning objects horizontally*

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:

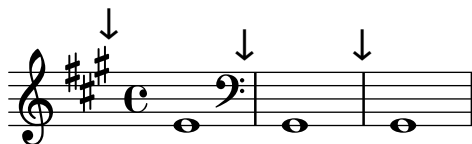
```
% The rehearsal mark will be aligned to the right edge of the Clef
\override Score.RehearsalMark.break-align-symbols = #'(clef)
\key a \major
\clef treble
\mark "↓"
e'1
% The rehearsal mark will be aligned to the left edge of the Time Signature
\override Score.RehearsalMark.break-align-symbols = #'(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
\breathe
\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.

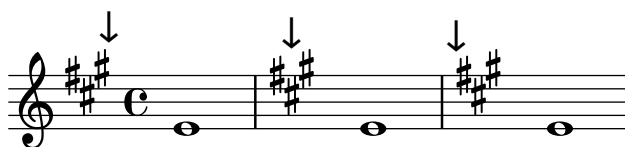
```
% The rehearsal mark will be aligned to the right edge of the Key Signature
\override Score.RehearsalMark.break-align-symbols = #'(key-signature clef)
\key a \major
\clef treble
\mark "↓"
e'1
% The rehearsal mark will be aligned to the right edge of the Clef
\set Staff.explicitKeySignatureVisibility = #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
```

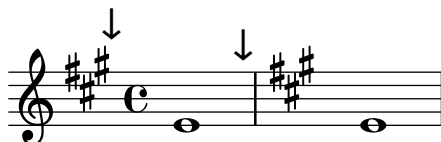


The rehearsal mark can also be offset to the right or left of the left edge by an arbitrary amount. The units are staff-spaces:

```
% 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], pàgina 245.

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 Secció A.8 [The Feta font], pàgina 651.

EPS files and Postscript commands can both be inserted inline using the `\epsfile` and `\postscript` markup commands respectively – see Secció A.11.3 [Graphic], pàgina 699.

## Vegeu també

Notation Reference: [Graphic notation inside markup], pàgina 245, Secció 1.8.2 [Formatting text], pàgina 237, Secció A.11 [Text markup commands], pàgina 674, Secció A.8 [The Feta font], pàgina 651, Secció A.11.3 [Graphic], pàgina 699.

### 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\)
}
```

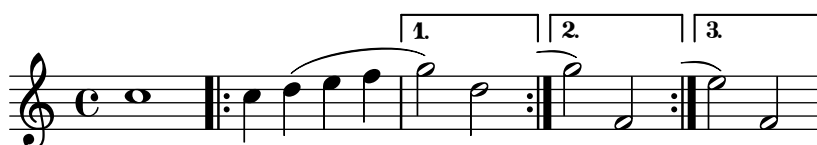


### *Specifying control points explicitly*

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,
}
```



### Advertiments i problemes coneguts

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.

### Vegeu també

Internals Reference: Secció “TieColumn” in *Referència de funcionament intern*.

#### 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.

## Advertiments i problemes coneguts

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`.

## Vegeu també

Extending LilyPond: Secció “Difficult tweaks” in *Extendre*.

### 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 Using music functions

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 Secció “Music functions” in *Extendre*.

### 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

|                    |                                                                                                                                                                                                                                                                 |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>argN</i>        | <i>nth</i> argument                                                                                                                                                                                                                                             |
| <i>typeN?</i>      | a scheme <i>type predicate</i> for which <i>argN</i> must return <code>#t</code> .                                                                                                                                                                              |
| <i>...music...</i> | normal LilyPond input, using <code>\$</code> (in places where only LilyPond constructs are allowed) or <code>#</code> (to use it as a Scheme value or music function argument or music inside of music lists) to reference arguments (eg. <code>#arg1</code> ). |

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 Secció A.21 [Predefined type predicates], pàgina 784. User-defined type predicates are also allowed.

## Vegeu també

Notation Reference: Secció A.21 [Predefined type predicates], pàgina 784.

Extending LilyPond: Secció “Music functions” in *Extendre*.

Installed Files: `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
```

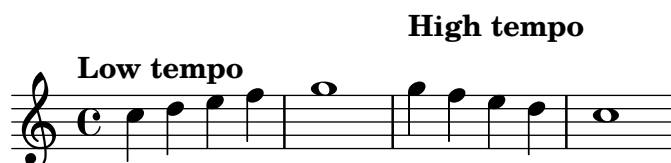
```
\relative { c'4 d e f \custosNote g }
```



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
}
```



## Annex A Notation manual tables

### A.1 Chord name chart

The following chart shows two standard systems for printing chord names, along with the pitches they represent.

|                    |                        |                      |                       |                       |
|--------------------|------------------------|----------------------|-----------------------|-----------------------|
| 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

Alt

C<sup>7</sup>  $\flat 9$   $\flat 13$

C<sup>11</sup>  $\flat 9$   $\flat 13$

C<sup>7</sup>  $\sharp 11$

C<sup>9</sup>  $\sharp 11$

C $\triangle 9$

C<sup>9</sup>  $\sharp 7$

C<sup>7</sup>  $\flat 13$

C<sup>11</sup>  $\flat 13$

Def

Alt

C<sup>7</sup>  $\flat 9$   $\flat 13$

C<sup>11</sup>  $\flat 9$   $\flat 13$

C<sup>7</sup>  $\flat 9$  13

C<sup>13</sup>  $\flat 9$

C $\triangle 9$

C<sup>9</sup>  $\sharp 7$

C $\triangle 13$

C<sup>13</sup>  $\sharp 7$

Def

Alt

C $\triangle \sharp 11$

C<sup>9</sup>  $\sharp 7$   $\sharp 11$

C<sup>7</sup>  $\flat 9$  13

C<sup>13</sup>  $\flat 9$

C<sup>sus4</sup>

C add4 5

C<sup>7</sup> <sup>sus4</sup>

C add4 5 7

Def

Alt

C<sup>9</sup> <sup>sus4</sup>

C add4 5 7 9

C<sup>9</sup>

C add9

Cm<sup>11</sup>

C $\flat 3$  add11

Def

Alt

C<sup>lyd</sup>

C $\sharp 7$  add $\sharp 11$




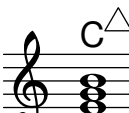




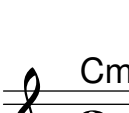
C<sup>alt</sup>




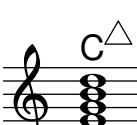


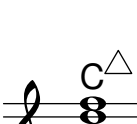
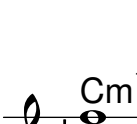
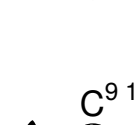
C<sup>7</sup>  $\flat 9$   $\flat 10$   $\sharp 11$   $\flat 13$

A.2 Common chord modifiers

The following table shows chord modifiers that can be used to generate standard chord structures.

| Type  | Interval                      | Modifier     | Example | Output |
|-------|-------------------------------|--------------|---------|--------|
| Major | Major third,<br>perfect fifth | 5 or nothing | c1:5    |        |
| Minor | Minor third,<br>perfect fifth | m or m5      | c1:m    |        |

|                         |                                         |             |          |                                                                                       |
|-------------------------|-----------------------------------------|-------------|----------|---------------------------------------------------------------------------------------|
| Augmented               | Major third,<br>augmented fifth         | aug         | c1:aug   |    |
| Diminished              | Minor third,<br>diminished fifth        | dim         | c1:dim   |    |
| Dominant seventh        | Major triad,<br>minor seventh           | 7           | c1:7     |    |
| Major seventh           | Major triad,<br>major seventh           | maj7 or maj | c1:maj7  |    |
| Minor seventh           | Minor triad,<br>minor seventh           | m7          | c1:m7    |  |
| Diminished seventh      | Diminished triad,<br>diminished seventh | dim7        | c1:dim7  |  |
| Augmented seventh       | Augmented triad,<br>minor seventh       | aug7        | c1:aug7  |  |
| Half-diminished seventh | Diminished triad,<br>minor seventh      | m7.5-       | c1:m7.5- |  |
| Minor-major seventh     | Minor triad,<br>major seventh           | m7+         | c1:m7+   |  |

|                     |                                     |       |          |                                                                                       |
|---------------------|-------------------------------------|-------|----------|---------------------------------------------------------------------------------------|
| Major sixth         | Major triad,<br>sixth               | 6     | c1:6     |    |
| Minor sixth         | Minor triad,<br>sixth               | m6    | c1:m6    |    |
| Dominant ninth      | Dominant seventh,<br>major ninth    | 9     | c1:9     |    |
| Major ninth         | Major seventh,<br>major ninth       | maj9  | c1:maj9  |   |
| Minor ninth         | Minor seventh,<br>major ninth       | m9    | c1:m9    |  |
| Dominant eleventh   | Dominant ninth,<br>perfect eleventh | 11    | c1:11    |  |
| Major eleventh      | Major ninth,<br>perfect eleventh    | maj11 | c1:maj11 |  |
| Minor eleventh      | Minor ninth,<br>perfect eleventh    | m11   | c1:m11   |  |
| Dominant thirteenth | Dominant ninth,<br>major thirteenth | 13    | c1:13    |  |

|                               |                                        |          |                        |  |
|-------------------------------|----------------------------------------|----------|------------------------|--|
| Dominant<br>thirteenth        | Dominant eleventh,<br>major thirteenth | 13.11    | c1:13.11               |  |
| Major thirteenth              | Major eleventh,<br>major thirteenth    | maj13.11 | c1:maj13.11            |  |
| Minor thirteenth              | Minor eleventh,<br>major thirteenth    | m13.11   | c1:m13.11              |  |
| Suspended second              | Major second,<br>perfect fifth         | sus2     | c1:sus2                |  |
| Suspended fourth              | Perfect fourth,<br>perfect fifth       | sus4     | c1:sus4                |  |
| Power chord<br>(two-voiced)   | Perfect fifth                          | 1.5      | \powerChords<br>c1:5   |  |
| Power chord<br>(three-voiced) | Perfect fifth,<br>octave               | 1.5.8    | \powerChords<br>c1:5.8 |  |

A.3 Predefined string tunings

The chart below shows the predefined string tunings.

Guitar tunings



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-five-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

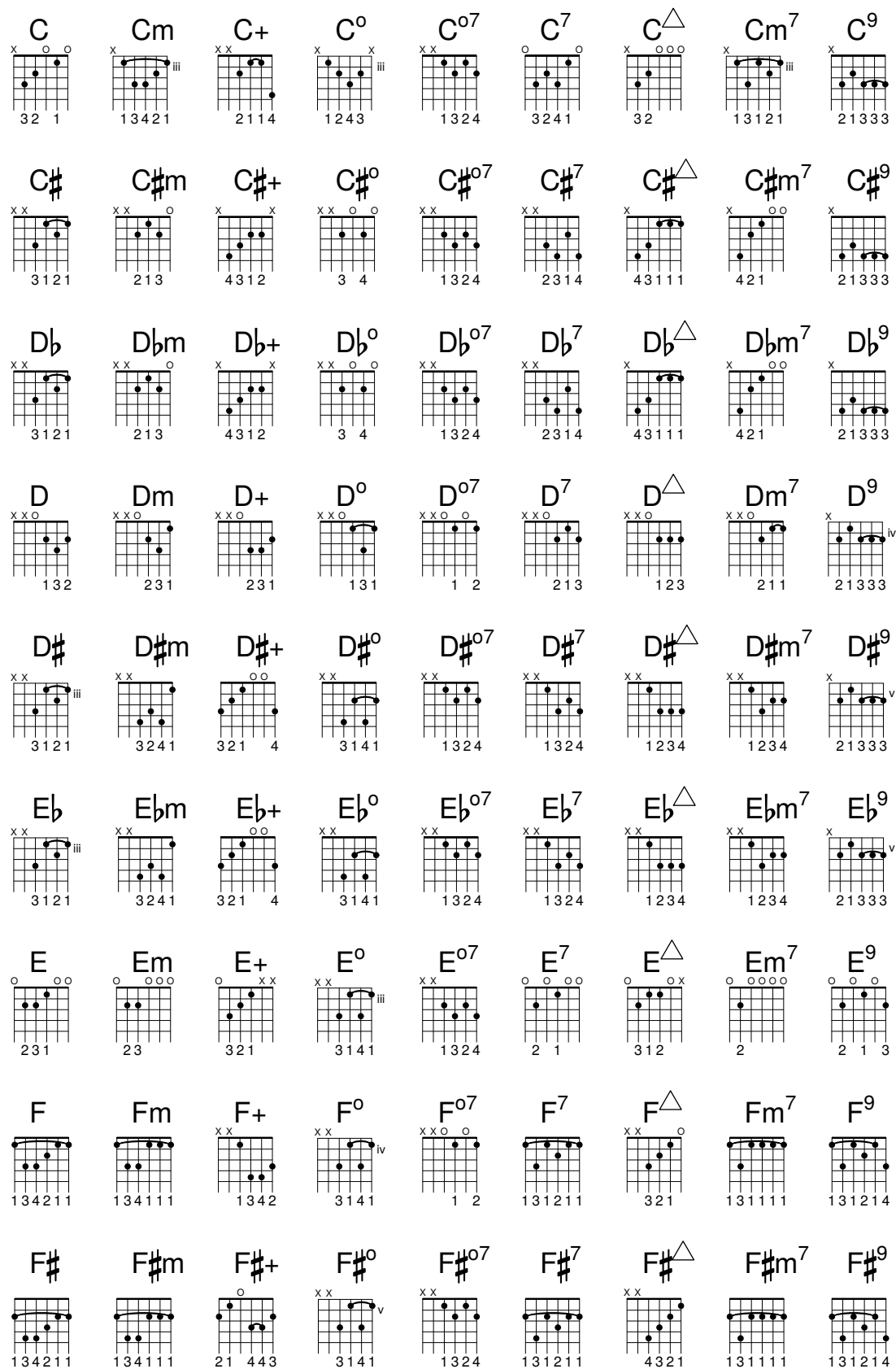
21 **Ukulele tunings**  
ukulele-tuning ukulele-d-tuning

23 tenor-ukulele-tuning baritone-ukulele-tuning

25 **Orchestral string tunings**  
violin-tuning viola-tuning cello-tuning double-bass-tuning

## A.4 Predefined fretboard diagrams

### Diagrams for Guitar



|                              |                                |                              |                                  |                               |                                |                                   |                                  |                                |
|------------------------------|--------------------------------|------------------------------|----------------------------------|-------------------------------|--------------------------------|-----------------------------------|----------------------------------|--------------------------------|
| $G\flat$<br><br>1 3 4 2 1 1  | $G\flat m$<br><br>1 3 4 1 1 1  | $G\flat +$<br><br>2 1 4 4 3  | $G\flat^{\circ}$<br><br>3 1 4 1  | $G\flat^{o7}$<br><br>1 3 2 4  | $G\flat^7$<br><br>1 3 1 2 1 1  | $G\flat^{\Delta}$<br><br>4 3 2 1  | $G\flat m^7$<br><br>1 3 1 1 1 1  | $G\flat^9$<br><br>1 3 1 2 1 4  |
| $G$<br><br>2 1 3             | $Gm$<br><br>1 3 4 1 1 1        | $G+$<br><br>1 3 4 2          | $G^{\circ}$<br><br>3 1 4 1       | $G^{o7}$<br><br>1 3 2 4       | $G^7$<br><br>3 2 1             | $G^{\Delta}$<br><br>4 3 2 1       | $Gm^7$<br><br>1 3 1 1 1 1        | $G^9$<br><br>1 3 1 2 1 4       |
| $G\sharp$<br><br>1 3 4 2 1 1 | $G\sharp m$<br><br>1 3 4 1 1 1 | $G\sharp +$<br><br>4 3 1 2   | $G\sharp^{\circ}$<br><br>3 1 4 1 | $G\sharp^{o7}$<br><br>1 2     | $G\sharp^7$<br><br>1 3 1 2 1 1 | $G\sharp^{\Delta}$<br><br>1 1 1 3 | $G\sharp m^7$<br><br>1 3 1 1 1 1 | $G\sharp^9$<br><br>1 3 1 2 1 4 |
| $A\flat$<br><br>1 3 4 2 1 1  | $A\flat m$<br><br>1 3 4 1 1 1  | $A\flat +$<br><br>4 3 1 2    | $A\flat^{\circ}$<br><br>3 1 4 1  | $A\flat^{o7}$<br><br>1 2      | $A\flat^7$<br><br>1 3 1 2 1 1  | $A\flat^{\Delta}$<br><br>1 1 1 3  | $A\flat m^7$<br><br>1 3 1 1 1 1  | $A\flat^9$<br><br>1 3 1 2 1 4  |
| $A$<br><br>1 2 3             | $Am$<br><br>2 3 1              | $A+$<br><br>4 2 3 1          | $A^{\circ}$<br><br>1 2 3         | $A^{o7}$<br><br>1 3 2 4       | $A^7$<br><br>1 3               | $A^{\Delta}$<br><br>2 1 3         | $Am^7$<br><br>2 1                | $A^9$<br><br>1 3 1 2 1 4       |
| $A\sharp$<br><br>1 2 3 4 1   | $A\sharp m$<br><br>1 3 4 2 1   | $A\sharp +$<br><br>2 1 4 4 3 | $A\sharp^{\circ}$<br><br>1 2 4 3 | $A\sharp^{o7}$<br><br>1 3 2 4 | $A\sharp^7$<br><br>1 2 1 3 1   | $A\sharp^{\Delta}$<br><br>1 3 2 4 | $A\sharp m^7$<br><br>1 3 1 2 1   | $A\sharp^9$<br><br>1 3 1 2 1 4 |
| $B\flat$<br><br>1 2 3 4 1    | $B\flat m$<br><br>1 3 4 2 1    | $B\flat +$<br><br>2 1 4 4 3  | $B\flat^{\circ}$<br><br>1 2 4 3  | $B\flat^{o7}$<br><br>1 3 2 4  | $B\flat^7$<br><br>1 2 1 3 1    | $B\flat^{\Delta}$<br><br>1 3 2 4  | $B\flat m^7$<br><br>1 3 1 2 1    | $B\flat^9$<br><br>1 3 1 2 1 4  |
| $B$<br><br>1 2 3 4 1         | $Bm$<br><br>1 3 4 2 1          | $B+$<br><br>2 1              | $B^{\circ}$<br><br>1 2 4 3       | $B^{o7}$<br><br>1 2           | $B^7$<br><br>2 1 3 4           | $B^{\Delta}$<br><br>1 3 2 4       | $Bm^7$<br><br>1 3 1 2 1          | $B^9$<br><br>2 1 3 3 3         |

## Diagrams for Ukulele

|              |                   |                 |                            |                |                       |                       |                    |                       |                       |                  |
|--------------|-------------------|-----------------|----------------------------|----------------|-----------------------|-----------------------|--------------------|-----------------------|-----------------------|------------------|
| $C$<br><br>3 | $Cm$<br><br>1 2 3 | $C+$<br><br>1 4 | $C^{\circ}$<br><br>1 3 2 4 | $C^7$<br><br>1 | $C^{\Delta}$<br><br>1 | $Cm^7$<br><br>1 1 1 1 | $C^6$<br><br>1 2 2 | $C^{sus2}$<br><br>1 3 | $C^{sus4}$<br><br>2 1 | $C^9$<br><br>2 1 |
|--------------|-------------------|-----------------|----------------------------|----------------|-----------------------|-----------------------|--------------------|-----------------------|-----------------------|------------------|

|            |              |              |                  |              |                      |                |              |                   |                   |              |
|------------|--------------|--------------|------------------|--------------|----------------------|----------------|--------------|-------------------|-------------------|--------------|
| $C^\sharp$ | $C^\sharp m$ | $C^\sharp +$ | $C^\sharp^\circ$ | $C^\sharp^7$ | $C^\sharp^\triangle$ | $C^\sharp m^7$ | $C^\sharp^6$ | $C^\sharp^{sus2}$ | $C^\sharp^{sus4}$ | $C^\sharp^9$ |
|            |              |              |                  |              |                      |                |              |                   |                   |              |
| 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      |
| $D^\flat$  | $D^\flat m$  | $D^\flat +$  | $D^\flat^\circ$  | $D^\flat^7$  | $D^\flat^\triangle$  | $D^\flat m^7$  | $D^\flat^6$  | $D^\flat^{sus2}$  | $D^\flat^{sus4}$  | $D^\flat^9$  |
|            |              |              |                  |              |                      |                |              |                   |                   |              |
| 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      |
| $D$        | $Dm$         | $D+$         | $D^\circ$        | $D^7$        | $D^\triangle$        | $Dm^7$         | $D^6$        | $D^{sus2}$        | $D^{sus4}$        | $D^9$        |
|            |              |              |                  |              |                      |                |              |                   |                   |              |
| 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      |
| $D^\sharp$ | $D^\sharp m$ | $D^\sharp +$ | $D^\sharp^\circ$ | $D^\sharp^7$ | $D^\sharp^\triangle$ | $D^\sharp m^7$ | $D^\sharp^6$ | $D^\sharp^{sus2}$ | $D^\sharp^{sus4}$ | $D^\sharp^9$ |
|            |              |              |                  |              |                      |                |              |                   |                   |              |
| 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        |
| $E^\flat$  | $E^\flat m$  | $E^\flat +$  | $E^\flat^\circ$  | $E^\flat^7$  | $E^\flat^\triangle$  | $E^\flat m^7$  | $E^\flat^6$  | $E^\flat^{sus2}$  | $E^\flat^{sus4}$  | $E^\flat^9$  |
|            |              |              |                  |              |                      |                |              |                   |                   |              |
| 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        |
| $E$        | $Em$         | $E+$         | $E^\circ$        | $E^7$        | $E^\triangle$        | $Em^7$         | $E^6$        | $E^{sus2}$        | $E^{sus4}$        | $E^9$        |
|            |              |              |                  |              |                      |                |              |                   |                   |              |
| 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      |
| $F$        | $Fm$         | $F+$         | $F^\circ$        | $F^7$        | $F^\triangle$        | $Fm^7$         | $F^6$        | $F^{sus2}$        | $F^{sus4}$        | $F^9$        |
|            |              |              |                  |              |                      |                |              |                   |                   |              |
| 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      |
| $F^\sharp$ | $F^\sharp m$ | $F^\sharp +$ | $F^\sharp^\circ$ | $F^\sharp^7$ | $F^\sharp^\triangle$ | $F^\sharp m^7$ | $F^\sharp^6$ | $F^\sharp^{sus2}$ | $F^\sharp^{sus4}$ | $F^\sharp^9$ |
|            |              |              |                  |              |                      |                |              |                   |                   |              |
| 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      |
| $G^\flat$  | $G^\flat m$  | $G^\flat +$  | $G^\flat^\circ$  | $G^\flat^7$  | $G^\flat^\triangle$  | $G^\flat m^7$  | $G^\flat^6$  | $G^\flat^{sus2}$  | $G^\flat^{sus4}$  | $G^\flat^9$  |
|            |              |              |                  |              |                      |                |              |                   |                   |              |
| 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      |
| $G$        | $Gm$         | $G+$         | $G^\circ$        | $G^7$        | $G^\triangle$        | $Gm^7$         | $G^6$        | $G^{sus2}$        | $G^{sus4}$        | $G^9$        |
|            |              |              |                  |              |                      |                |              |                   |                   |              |
| 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      |

|            |              |              |                  |              |                      |                |              |                        |                        |              |
|------------|--------------|--------------|------------------|--------------|----------------------|----------------|--------------|------------------------|------------------------|--------------|
| $G^\sharp$ | $G^\sharp m$ | $G^\sharp +$ | $G^\sharp \circ$ | $G^\sharp 7$ | $G^\sharp \triangle$ | $G^\sharp m^7$ | $G^\sharp 6$ | $G^\sharp \text{sus}2$ | $G^\sharp \text{sus}4$ | $G^\sharp 9$ |
|            |              |              |                  |              |                      |                |              |                        |                        |              |
| 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        |
| $A^\flat$  | $A^\flat m$  | $A^\flat +$  | $A^\flat \circ$  | $A^\flat 7$  | $A^\flat \triangle$  | $A^\flat m^7$  | $A^\flat 6$  | $A^\flat \text{sus}2$  | $A^\flat \text{sus}4$  | $A^\flat 9$  |
|            |              |              |                  |              |                      |                |              |                        |                        |              |
| 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        |
| $A$        | $A m$        | $A +$        | $A \circ$        | $A 7$        | $A \triangle$        | $A m^7$        | $A 6$        | $A \text{sus}2$        | $A \text{sus}4$        | $A 9$        |
|            |              |              |                  |              |                      |                |              |                        |                        |              |
| 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          |
| $A^\sharp$ | $A^\sharp m$ | $A^\sharp +$ | $A^\sharp \circ$ | $A^\sharp 7$ | $A^\sharp \triangle$ | $A^\sharp m^7$ | $A^\sharp 6$ | $A^\sharp \text{sus}2$ | $A^\sharp \text{sus}4$ | $A^\sharp 9$ |
|            |              |              |                  |              |                      |                |              |                        |                        |              |
| 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      |
| $B^\flat$  | $B^\flat m$  | $B^\flat +$  | $B^\flat \circ$  | $B^\flat 7$  | $B^\flat \triangle$  | $B^\flat m^7$  | $B^\flat 6$  | $B^\flat \text{sus}2$  | $B^\flat \text{sus}4$  | $B^\flat 9$  |
|            |              |              |                  |              |                      |                |              |                        |                        |              |
| 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      |
| $B$        | $B m$        | $B +$        | $B \circ$        | $B 7$        | $B \triangle$        | $B m^7$        | $B 6$        | $B \text{sus}2$        | $B \text{sus}4$        | $B 9$        |
|            |              |              |                  |              |                      |                |              |                        |                        |              |
| 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      |

## Diagrams for Mandolin

|            |              |              |                      |              |                      |                |                      |              |                        |                        |              |
|------------|--------------|--------------|----------------------|--------------|----------------------|----------------|----------------------|--------------|------------------------|------------------------|--------------|
| $C$        | $C m$        | $C +$        | $C^{\circ 7}$        | $C 7$        | $C \triangle$        | $C m^7$        | $C^\emptyset$        | $C 6$        | $C \text{sus}2$        | $C \text{sus}4$        | $C 9$        |
|            |              |              |                      |              |                      |                |                      |              |                        |                        |              |
| 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        |
| $C^\sharp$ | $C^\sharp m$ | $C^\sharp +$ | $C^\sharp^{\circ 7}$ | $C^\sharp 7$ | $C^\sharp \triangle$ | $C^\sharp m^7$ | $C^\sharp^\emptyset$ | $C^\sharp 6$ | $C^\sharp \text{sus}2$ | $C^\sharp \text{sus}4$ | $C^\sharp 9$ |
|            |              |              |                      |              |                      |                |                      |              |                        |                        |              |
| 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      |
| $D^\flat$  | $D^\flat m$  | $D^\flat +$  | $D^\flat^{\circ 7}$  | $D^\flat 7$  | $D^\flat \triangle$  | $D^\flat m^7$  | $D^\flat^\emptyset$  | $D^\flat 6$  | $D^\flat \text{sus}2$  | $D^\flat \text{sus}4$  | $D^\flat 9$  |
|            |              |              |                      |              |                      |                |                      |              |                        |                        |              |
| 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            |

|           |             |            |                |             |                    |               |                       |             |                  |                  |             |
|-----------|-------------|------------|----------------|-------------|--------------------|---------------|-----------------------|-------------|------------------|------------------|-------------|
| $A\flat$  | $A\flat m$  | $A\flat+$  | $A\flat^{o7}$  | $A\flat^7$  | $A\flat^{\Delta}$  | $A\flat m^7$  | $A\flat^{\emptyset}$  | $A\flat^6$  | $A\flat^{sus2}$  | $A\flat^{sus4}$  | $A\flat^9$  |
|           |             |            |                |             |                    |               |                       |             |                  |                  |             |
| 1 1 3 4   | 1 1 2 4     | 1 2 3 4    | 1 3 2          | 1 1 3 2     | 1 1 3 3            | 1 1 2 2       | 1 2 2                 | 1 1 3 1     | 1 1 1 4          | 1 1 3 4          | 1 3 2 4     |
| $A$       | $A m$       | $A+$       | $A^{o7}$       | $A^7$       | $A^{\Delta}$       | $A m^7$       | $A^{\emptyset}$       | $A^6$       | $A^{sus2}$       | $A^{sus4}$       | $A^9$       |
|           |             |            |                |             |                    |               |                       |             |                  |                  |             |
| 1 1 3     | 1 1 2       | 2 3 4 1    | 2 1 4 3        | 1 1 3 2     | 1 1 3 3            | 1 1 2 2       | 2 1 3 4               | 1 1 3 1     | 1 1 1            | 1                | 1 3 2 4     |
| $A\sharp$ | $A\sharp m$ | $A\sharp+$ | $A\sharp^{o7}$ | $A\sharp^7$ | $A\sharp^{\Delta}$ | $A\sharp m^7$ | $A\sharp^{\emptyset}$ | $A\sharp^6$ | $A\sharp^{sus2}$ | $A\sharp^{sus4}$ | $A\sharp^9$ |
|           |             |            |                |             |                    |               |                       |             |                  |                  |             |
| 3 1 1     | 1 1 2 4     | 3 1 2      | 2 1 4 3        | 1 1 3 2     | 3 1                | 1 1 2 2       | 2 1 3 4               | 1 1         | 1 1 1 4          | 3 1 1 1          | 1 2 3       |
| $B\flat$  | $B\flat m$  | $B\flat+$  | $B\flat^{o7}$  | $B\flat^7$  | $B\flat^{\Delta}$  | $B\flat m^7$  | $B\flat^{\emptyset}$  | $B\flat^6$  | $B\flat^{sus2}$  | $B\flat^{sus4}$  | $B\flat^9$  |
|           |             |            |                |             |                    |               |                       |             |                  |                  |             |
| 3 1 1     | 1 1 2 4     | 3 1 2      | 2 1 4 3        | 1 1 3 2     | 3 1                | 1 1 2 2       | 2 1 3 4               | 1 1         | 1 1 1 4          | 3 1 1 1          | 1 2 3       |
| $B$       | $B m$       | $B+$       | $B^{o7}$       | $B^7$       | $B^{\Delta}$       | $B m^7$       | $B^{\emptyset}$       | $B^6$       | $B^{sus2}$       | $B^{sus4}$       | $B^9$       |
|           |             |            |                |             |                    |               |                       |             |                  |                  |             |
| 1 1 3 4   | 3 1 1       | 4 1 2 3    | 1 3 2          | 1 1 3 2     | 4 1 1 2            | 3 1           | 2 3 1                 | 1 1 2 2     | 1 1 1 4          | 3 1 1 1          | 2 1 3 4     |

## A.5 Predefined paper sizes

Paper sizes are defined in `scm/paper.scm`

### The “ISO 216” A Series

|       |                 |
|-------|-----------------|
| "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) |

### The “ISO 216” B Series

|       |              |
|-------|--------------|
| "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) |

**Two extended sizes as defined in “DIN 476”**

|       |                  |
|-------|------------------|
| "4a0" | (1682 x 2378 mm) |
| "2a0" | (1189 x 1682 mm) |

**“ISO 269” standard C series**

|       |                 |
|-------|-----------------|
| "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) |

**North American paper sizes**

|                |                  |
|----------------|------------------|
| "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 by IEEE Printer Working Group, for children’s writing**

|                     |               |
|---------------------|---------------|
| "government-letter" | (8 x 10.5 in) |
|---------------------|---------------|



"government-legal"  
(8.5 x 13.0 in)

"philippine-legal"  
(8.5 x 13.0 in)

#### **ANSI sizes**

"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)

#### **North American Architectural sizes**

"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)

#### **Antique sizes still used in the United Kingdom**

"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)

#### **PA4 based sizes**

"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)

#### **Used in Southeast Asia and Australia**

"f4" (210 x 330 mm)

**Used for very small @lilypond examples in the documentation based on a8 landscape.**

"a8landscape"  
(74 x 52 mm)

## A.6 MIDI instruments

The following is a list of names that can be used for the `midiInstrument` property. The order of the instruments below, starting in the left-hand column moving down, corresponds to the General MIDI Standard's 128 Program Numbers.

|                         |                   |                    |
|-------------------------|-------------------|--------------------|
| 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 List of colors

## Normal colors

Usage syntax is detailed in [Coloring objects], pàgina 222.

|          |             |            |          |
|----------|-------------|------------|----------|
| black    | white       | red        | green    |
| blue     | cyan        | magenta    | yellow   |
| grey     | darkred     | darkgreen  | darkblue |
| darkcyan | darkmagenta | darkyellow |          |

## X color names

X color names come several variants:

Any name that is spelled as a single word with capitalization (e.g., ‘LightSlateBlue’) can also be spelled as space separated words without capitalization (e.g., ‘light slate blue’).

The word ‘grey’ can always be spelled ‘gray’ (e.g., ‘DarkSlateGray’).

Some names can take a numerical suffix (e.g., ‘LightSalmon4’).

## Color Names without a numerical suffix:

|                |                      |                   |               |                  |
|----------------|----------------------|-------------------|---------------|------------------|
| 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     |                      |                   |               |                  |

## Color names with a numerical suffix

In the following names the suffix N can be a number in the range 1-4:

|                |               |               |              |            |
|----------------|---------------|---------------|--------------|------------|
| 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       |                 |              |                 |                |

## Grey Scale

A grey scale can be obtained using:

`greyN`

Where N is in the range 0-100.

## A.8 The Feta font

The following symbols are available in the Emmentaler font and may be accessed directly using text markup with the name of the glyph as shown in the tables below, such as `g^\markup {\musicglyph #"scripts.segno" }` or `\markup {\musicglyph #"five"}`. For more information, see Secció 1.8.2 [Formatting text], pàgina 237.

### Clef glyphs

`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<br/>.varpercussion_change</code> | <b>□</b>                                                                                  |
| <code>clefs.tab</code>           | <b><math>\mathcal{T}</math><br/><math>\mathcal{A}</math><br/><math>\mathcal{B}</math></b> | <code>clefs.tab_change</code>                | <b><math>\mathcal{T}</math><br/><math>\mathcal{A}</math><br/><math>\mathcal{B}</math></b> |

## Time Signature glyphs



















|                          |          |                          |          |
|--------------------------|----------|--------------------------|----------|
| <code>timesig.C44</code> | <b>C</b> | <code>timesig.C22</code> | <b>Ⓒ</b> |
|--------------------------|----------|--------------------------|----------|

## Number glyphs







|                     |          |                     |          |
|---------------------|----------|---------------------|----------|
| <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>five</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> |

## Accidental glyphs



|                                                     |           |                                                              |           |
|-----------------------------------------------------|-----------|--------------------------------------------------------------|-----------|
| <code>accidentals.sharp</code>                      | <b>#</b>  | <code>accidentals<br/>.sharp.arrowup</code>                  | <b>↗#</b> |
| <code>accidentals<br/>.sharp.arrowdown</code>       | <b>↘#</b> | <code>accidentals<br/>.sharp.arrowboth</code>                | <b>↕#</b> |
| <code>accidentals.sharp<br/>.slashslash.stem</code> | <b>‡</b>  | <code>accidentals.sharp<br/>.slashslashslash.stemstem</code> | <b>‡‡</b> |

|                                                                      |                                                                                     |                                                                         |                                                                                       |
|----------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| <code>accidentals.sharp</code><br><code>.slashslashslash.stem</code> |    | <code>accidentals.sharp</code><br><code>.slashslash.stemstemstem</code> |    |
| <code>accidentals.doublsharp</code>                                  |    | <code>accidentals.natural</code>                                        |    |
| <code>accidentals</code><br><code>.natural.arrowup</code>            |    | <code>accidentals</code><br><code>.natural.arrowdown</code>             |    |
| <code>accidentals</code><br><code>.natural.arrowboth</code>          |    | <code>accidentals.flat</code>                                           |    |
| <code>accidentals.flat.arrowup</code>                                |    | <code>accidentals</code><br><code>.flat.arrowdown</code>                |    |
| <code>accidentals</code><br><code>.flat.arrowboth</code>             |   | <code>accidentals.flat.slash</code>                                     |    |
| <code>accidentals.flat</code><br><code>.slashslash</code>            |  | <code>accidentals</code><br><code>.mirroredflat.flat</code>             |  |
| <code>accidentals.mirroredflat</code>                                |  | <code>accidentals</code><br><code>.mirroredflat.backslash</code>        |  |
| <code>accidentals.flatflat</code>                                    |  | <code>accidentals</code><br><code>.flatflat.slash</code>                |  |
| <code>accidentals.rightparen</code>                                  | )                                                                                   | <code>accidentals.leftparen</code>                                      | (                                                                                     |


## Default Notehead glyphs

|                            |                                                                                     |                            |                                                                                       |
|----------------------------|-------------------------------------------------------------------------------------|----------------------------|---------------------------------------------------------------------------------------|
| <code>noteheads.uM2</code> |  | <code>noteheads.dM2</code> |  |
| <code>noteheads.sM1</code> |  | <code>noteheads.s0</code>  |  |
| <code>noteheads.s1</code>  |  | <code>noteheads.s2</code>  |  |

## Special Notehead glyphs




















|                                   |                                                                                     |                                   |                                                                                       |
|-----------------------------------|-------------------------------------------------------------------------------------|-----------------------------------|---------------------------------------------------------------------------------------|
| <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.ultriangle</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> |  |

## Shape-note Notehead glyphs

|                                 |                                                                                     |                                 |                                                                                       |
|---------------------------------|-------------------------------------------------------------------------------------|---------------------------------|---------------------------------------------------------------------------------------|
| <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> |  |



|                                   |                                                                                     |                                   |                                                                                       |
|-----------------------------------|-------------------------------------------------------------------------------------|-----------------------------------|---------------------------------------------------------------------------------------|
| <code>noteheads.s0re</code>       |    | <code>noteheads.ulre</code>       |    |
| <code>noteheads.d1re</code>       |    | <code>noteheads.u2re</code>       |    |
| <code>noteheads.d2re</code>       |    | <code>noteheads.s0reThin</code>   |    |
| <code>noteheads.ulreThin</code>   |    | <code>noteheads.d1reThin</code>   |    |
| <code>noteheads.u2reThin</code>   |    | <code>noteheads.d2reThin</code>   |    |
| <code>noteheads.s0mi</code>       |    | <code>noteheads.s1mi</code>       |    |
| <code>noteheads.s2mi</code>       |   | <code>noteheads.s0miMirror</code> |   |
| <code>noteheads.s1miMirror</code> |  | <code>noteheads.s2miMirror</code> |  |
| <code>noteheads.s0miThin</code>   |  | <code>noteheads.s1miThin</code>   |  |
| <code>noteheads.s2miThin</code>   |  | <code>noteheads.u0fa</code>       |  |
| <code>noteheads.d0fa</code>       |  | <code>noteheads.u1fa</code>       |  |
| <code>noteheads.d1fa</code>       |  | <code>noteheads.u2fa</code>       |  |
| <code>noteheads.d2fa</code>       |  | <code>noteheads.u0faThin</code>   |  |
| <code>noteheads.d0faThin</code>   |  | <code>noteheads.u1faThin</code>   |  |
| <code>noteheads.d1faThin</code>   |  | <code>noteheads.u2faThin</code>   |  |








|                                 |                                                                                     |                                 |                                                                                       |
|---------------------------------|-------------------------------------------------------------------------------------|---------------------------------|---------------------------------------------------------------------------------------|
| <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 | ▴ |













|                      |   |                      |   |
|----------------------|---|----------------------|---|
| noteheads.s0reWalker | ◄ | noteheads.u1reWalker | ► |
| noteheads.d1reWalker | ◄ | noteheads.u2reWalker | ► |
| noteheads.d2reWalker | ◄ | noteheads.s0miWalker | ◊ |
| noteheads.s1miWalker | ◊ | noteheads.s2miWalker | ◊ |
| noteheads.s0faWalker | ▷ | noteheads.u1faWalker | ▷ |
| noteheads.d1faWalker | ▷ | noteheads.u2faWalker | ▷ |
| noteheads.d2faWalker | ▷ | noteheads.s0laWalker | ◻ |
| noteheads.s1laWalker | ◻ | noteheads.s2laWalker | ■ |
| noteheads.s0tiWalker | ◄ | noteheads.ultiWalker | ► |
| noteheads.d1tiWalker | ◄ | noteheads.u2tiWalker | ► |
| noteheads.d2tiWalker | ◄ |                      |   |

Rest glyphs

|          |   |           |   |
|----------|---|-----------|---|
| rests.0  | — | rests.1   | — |
| rests.0o | — | rests.1o  | — |
| rests.M3 |   | rests.M2  |   |
| rests.M1 | ■ | rests.M1o | ■ |

|          |                                                                                   |                  |                                                                                     |
|----------|-----------------------------------------------------------------------------------|------------------|-------------------------------------------------------------------------------------|
| rests.2  |  | rests.2classical |  |
| rests.2z |  | rests.3          |  |
| rests.4  |  | rests.5          |  |
| rests.6  |  | rests.7          |  |

Flag glyphs

|              |                                                                                     |              |                                                                                       |
|--------------|-------------------------------------------------------------------------------------|--------------|---------------------------------------------------------------------------------------|
| flags.u3     |    | flags.u4     |    |
| flags.u5     |  | flags.u6     |  |
| flags.u7     |  | flags.d3     |  |
| flags.d4     |  | flags.d5     |  |
| flags.d6     |  | flags.d7     |  |
| flags.ugrace |  | flags.dgrace |  |





















Dot glyphs































|          |                                                                                     |
|----------|-------------------------------------------------------------------------------------|
| dots.dot |  |
|----------|-------------------------------------------------------------------------------------|




















## Dynamic glyphs

|       |                 |   |                 |
|-------|-----------------|---|-----------------|
| 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> |   |                 |









## Script glyphs

|                          |                                                                                     |                          |                                                                                       |
|--------------------------|-------------------------------------------------------------------------------------|--------------------------|---------------------------------------------------------------------------------------|
| 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>         |    | <code>scripts.halfopenvertical</code>       |    |
| <code>scripts.stopped</code>          |    | <code>scripts.upbow</code>                  |    |
| <code>scripts.downbow</code>          |    | <code>scripts.reverseturn</code>            |    |
| <code>scripts.turn</code>             |    | <code>scripts.trill</code>                  |    |
| <code>scripts.upedalheel</code>       |    | <code>scripts.dpedalheel</code>             |    |
| <code>scripts.upedaltoe</code>        |    | <code>scripts.dpedaltoe</code>              |    |
| <code>scripts.flageolet</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<br/>.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><br><code>.usignumcongruentiae</code> |  | <code>scripts</code><br><code>.dsignumcongruentiae</code> |  |

## Arrowhead glyphs








|                                  |                                                                                     |                                   |                                                                                       |
|----------------------------------|-------------------------------------------------------------------------------------|-----------------------------------|---------------------------------------------------------------------------------------|
| <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> |  |



Bracket-tip glyphs

|                             |                                                                                   |                               |                                                                                     |
|-----------------------------|-----------------------------------------------------------------------------------|-------------------------------|-------------------------------------------------------------------------------------|
| <code>brackettips.up</code> |  | <code>brackettips.down</code> |  |
|-----------------------------|-----------------------------------------------------------------------------------|-------------------------------|-------------------------------------------------------------------------------------|

Pedal glyphs

|                        |                                                                                     |                      |                                                                                     |
|------------------------|-------------------------------------------------------------------------------------|----------------------|-------------------------------------------------------------------------------------|
| <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> |  |                      |                                                                                     |
































Accordion glyphs

|                                  |                                                                                     |                                |                                                                                       |
|----------------------------------|-------------------------------------------------------------------------------------|--------------------------------|---------------------------------------------------------------------------------------|
| <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>    |  |
















Tie glyphs

|                               |                                                                                     |                                 |                                                                                       |
|-------------------------------|-------------------------------------------------------------------------------------|---------------------------------|---------------------------------------------------------------------------------------|
| <code>ties.lyric.short</code> |  | <code>ties.lyric.default</code> |  |
|-------------------------------|-------------------------------------------------------------------------------------|---------------------------------|---------------------------------------------------------------------------------------|













## Vaticana glyphs





|                                                                       |                                                                                     |                                                                  |                                                                                       |
|-----------------------------------------------------------------------|-------------------------------------------------------------------------------------|------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| <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</code><br><code>.svaticana.punctum</code>        |  |
| <code>noteheads.svaticana</code><br><code>.punctum.cavum</code>       |  | <code>noteheads.svaticana</code><br><code>.linea.punctum</code>  |  |
| <code>noteheads.svaticana</code><br><code>.linea.punctum.cavum</code> |  | <code>noteheads.svaticana</code><br><code>.inclinatum</code>     |  |
| <code>noteheads.svaticana.lpes</code>                                 |  | <code>noteheads</code><br><code>.svaticana.vlpes</code>          |  |
| <code>noteheads.svaticana.upes</code>                                 |  | <code>noteheads</code><br><code>.svaticana.vupes</code>          |  |
| <code>noteheads</code><br><code>.svaticana.plica</code>               |  | <code>noteheads</code><br><code>.svaticana.vplica</code>         |  |
| <code>noteheads</code><br><code>.svaticana.epiphonus</code>           |  | <code>noteheads.svaticana</code><br><code>.vepiphonus</code>     |  |
| <code>noteheads.svaticana</code><br><code>.reverse.plica</code>       |  | <code>noteheads.svaticana</code><br><code>.reverse.vplica</code> |  |
| <code>noteheads.svaticana</code><br><code>.inner.cephalicus</code>    |  | <code>noteheads.svaticana</code><br><code>.cephalicus</code>     |  |
| <code>noteheads</code><br><code>.svaticana.quilisma</code>            |  |                                                                  |                                                                                       |

## Medicaea glyphs



























|                                               |                                                                                     |                                                  |                                                                                       |
|-----------------------------------------------|-------------------------------------------------------------------------------------|--------------------------------------------------|---------------------------------------------------------------------------------------|
| <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<br/>.inclinatum</code> |   |
| <code>noteheads<br/>.smedicaea.punctum</code> |  | <code>noteheads<br/>.smedicaea.rvirga</code>     |  |
| <code>noteheads<br/>.smedicaea.virga</code>   |  |                                                  |                                                                                       |

## Hufnagel glyphs

|                                   |                                                                                     |                                               |                                                                                       |
|-----------------------------------|-------------------------------------------------------------------------------------|-----------------------------------------------|---------------------------------------------------------------------------------------|
| <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<br/>.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<br>.shufnagel.punctum |  |
| noteheads<br>.shufnagel.virga |  | noteheads.shufnagel.lpes        |  |

## Mensural glyphs

|                       |                                                                                     |                                  |                                                                                       |
|-----------------------|-------------------------------------------------------------------------------------|----------------------------------|---------------------------------------------------------------------------------------|
| 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<br>.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           |  |
| flags.mensuralu03     |  | flags.mensuralu13                |  |



|                    |   |                    |   |
|--------------------|---|--------------------|---|
| flags.mensuralu23  | } | flags.mensurald03  | { |
| flags.mensurald13  | { | flags.mensurald23  | { |
| flags.mensuralu04  | } | flags.mensuralu14  | } |
| flags.mensuralu24  | } | flags.mensurald04  | { |
| flags.mensurald14  | { | flags.mensurald24  | { |
| flags.mensuralu05  | } | flags.mensuralu15  | } |
| flags.mensuralu25  | } | flags.mensurald05  | { |
| flags.mensurald15  | { | flags.mensurald25  | { |
| flags.mensuralu06  | } | flags.mensuralu16  | } |
| flags.mensuralu26  | } | flags.mensurald06  | { |
| flags.mensurald16  | { | flags.mensurald26  | { |
| timesig.mensural44 | C | timesig.mensural22 | ♢ |
| timesig.mensural32 | O | timesig.mensural64 | ⊙ |
| timesig.mensural94 | ⊙ | timesig.mensural34 | ϕ |

|                                   |   |                                  |   |
|-----------------------------------|---|----------------------------------|---|
| timesig.mensural68                | ¢ | timesig.mensural98               | 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<br>.srM3ligmensural     | ⏏ | noteheads.urM2mensural           | ⏏ |
| noteheads.drM2mensural            | ⏏ | noteheads<br>.srM2ligmensural    | ⏏ |
| noteheads.srM1mensural            | ⏏ | noteheads<br>.uM3semimensural    | ⏏ |
| noteheads<br>.dM3semimensural     | ⏏ | noteheads<br>.sM3semiligmensural | ⏏ |
| noteheads<br>.uM2semimensural     | ⏏ | noteheads<br>.dM2semimensural    | ⏏ |
| noteheads<br>.sM2semiligmensural  | ⏏ | noteheads<br>.sM1semimensural    | ⏏ |
| noteheads<br>.urM3semimensural    | ⏏ | noteheads<br>.drM3semimensural   | ⏏ |
| noteheads<br>.srM3semiligmensural | ⏏ | noteheads<br>.urM2semimensural   | ⏏ |











|                                   |                                                                                    |                                   |                                                                                      |
|-----------------------------------|------------------------------------------------------------------------------------|-----------------------------------|--------------------------------------------------------------------------------------|
| noteheads<br>.drM2semimensural    |   | noteheads<br>.srM2semiligmensural |   |
| noteheads<br>.srM1semimensural    |   | noteheads<br>.uM3blackmensural    |   |
| noteheads<br>.dM3blackmensural    |   | noteheads<br>.sM3blackligmensural |   |
| noteheads<br>.uM2blackmensural    |   | noteheads<br>.dM2blackmensural    |   |
| noteheads<br>.sM2blackligmensural |   | noteheads<br>.sM1blackmensural    |   |
| noteheads.s0mensural              |   | noteheads.s1mensural              |   |
| noteheads.s2mensural              |  | noteheads<br>.s0blackmensural     |  |

## Neomensural glyphs








|                       |                                                                                     |                                |                                                                                       |
|-----------------------|-------------------------------------------------------------------------------------|--------------------------------|---------------------------------------------------------------------------------------|
| rests.M3neomensural   |  | rests.M2neomensural            |  |
| rests.M1neomensural   |  | rests.0neomensural             |  |
| rests.1neomensural    |  | rests.2neomensural             |  |
| rests.3neomensural    |  | rests.4neomensural             |  |
| clefs.neomensural.c   |  | clefs.neomensural<br>.c_change |  |
| timesig.neomensural44 |  | timesig.neomensural22          |  |
| timesig.neomensural32 |  | timesig.neomensural64          |  |
| timesig.neomensural94 |  | timesig.neomensural34          |  |

|                                                         |                                                                                     |                                                         |                                                                                       |
|---------------------------------------------------------|-------------------------------------------------------------------------------------|---------------------------------------------------------|---------------------------------------------------------------------------------------|
| <code>timesig.neomensural68</code>                      |    | <code>timesig.neomensural98</code>                      |    |
| <code>timesig.neomensural48</code>                      |    | <code>timesig.neomensural68alt</code>                   |    |
| <code>timesig.neomensural24</code>                      |    | <code>noteheads.uM3neomensural</code>                   |    |
| <code>noteheads.dM3neomensural</code>                   |    | <code>noteheads.uM2neomensural</code>                   |    |
| <code>noteheads.dM2neomensural</code>                   |    | <code>noteheads.sM1neomensural</code>                   |    |
| <code>noteheads</code><br><code>.urM3neomensural</code> |    | <code>noteheads</code><br><code>.drM3neomensural</code> |    |
| <code>noteheads</code><br><code>.urM2neomensural</code> |   | <code>noteheads</code><br><code>.drM2neomensural</code> |   |
| <code>noteheads</code><br><code>.srM1neomensural</code> |  | <code>noteheads.s0neomensural</code>                    |  |
| <code>noteheads.s1neomensural</code>                    |  | <code>noteheads.s2neomensural</code>                    |  |








## Petrucchi glyphs

|                                 |                                                                                     |                                        |                                                                                       |
|---------------------------------|-------------------------------------------------------------------------------------|----------------------------------------|---------------------------------------------------------------------------------------|
| <code>clefs.petrucchi.c1</code> |  | <code>clefs.petrucchi.c1_change</code> |  |
| <code>clefs.petrucchi.c2</code> |  | <code>clefs.petrucchi.c2_change</code> |  |
| <code>clefs.petrucchi.c3</code> |  | <code>clefs.petrucchi.c3_change</code> |  |
| <code>clefs.petrucchi.c4</code> |  | <code>clefs.petrucchi.c4_change</code> |  |
| <code>clefs.petrucchi.c5</code> |  | <code>clefs.petrucchi.c5_change</code> |  |








|                                        |                                                                                   |                                        |                                                                                     |
|----------------------------------------|-----------------------------------------------------------------------------------|----------------------------------------|-------------------------------------------------------------------------------------|
| <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> |  |

## Solesmes glyphs

|                                              |                                                                                     |                                                |                                                                                       |
|----------------------------------------------|-------------------------------------------------------------------------------------|------------------------------------------------|---------------------------------------------------------------------------------------|
| <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>     |  |                                                |                                                                                       |

## Kievan Notation glyphs

|                                     |                                                                                     |                                     |                                                                                       |
|-------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------|---------------------------------------------------------------------------------------|
| <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>     |  |

|                                  |                                                                                   |                                 |                                                                                     |
|----------------------------------|-----------------------------------------------------------------------------------|---------------------------------|-------------------------------------------------------------------------------------|
| <code>noteheads.u2kievan</code>  |  | <code>noteheads.slkievan</code> |  |
| <code>noteheads.srlkievan</code> |  | <code>noteheads.d3kievan</code> |  |
| <code>noteheads.u3kievan</code>  |  |                                 |                                                                                     |

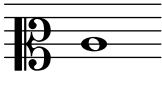


## A.9 Note head styles

The following styles may be used for note heads.

|                             |                             |
|-----------------------------|-----------------------------|
| <code>default</code>        | <code>altdefault</code>     |
| <code>baroque</code>        | <code>neomensural</code>    |
| <code>mensural</code>       | <code>petrucci</code>       |
| <code>harmonic</code>       | <code>harmonic-black</code> |
| <code>harmonic-mixed</code> | <code>diamond</code>        |
| <code>cross</code>          | <code>xcircle</code>        |
| <code>triangle</code>       | <code>slash</code>          |

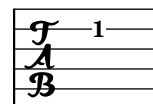
## A.10 Clef styles

The following table shows all the different clef styles possible (including where *middle C* sits relative to the clef).

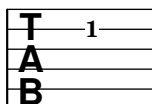
| Example                         | Output                                                                              | Example                        | Output                                                                                |
|---------------------------------|-------------------------------------------------------------------------------------|--------------------------------|---------------------------------------------------------------------------------------|
| <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 Text markup commands

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
 }
}

```

$E = mc^2$

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**

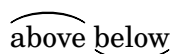
`\tie arg` (markup)

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 :
first right
second right

```

```

center-aligned :
left - - - - - right

```

```

left-aligned :
left: : : : : : : : : : : : : : : first
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 three
two

```

```

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 officia 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 officia  
 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
 }
}

```

}

[.]

`\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
 }
}
```

(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
}
```

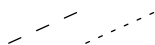


`\draw-dashed-line dest` (pair of numbers)

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)

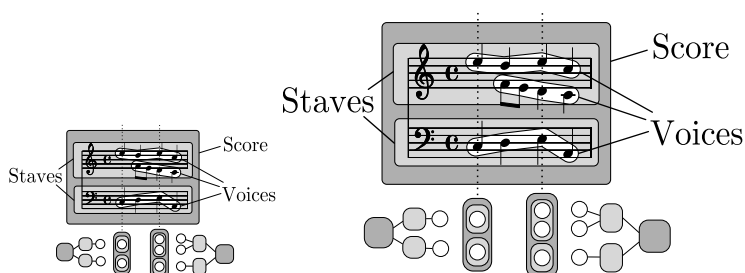


- `font-size` (0)
- `thickness` (1)

`\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
 }
}
```



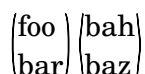
Used properties:

- `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
}
```

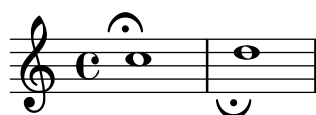
**##**

`\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 Secció

“The Feta font” in *Referència de la notació* 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.. 101

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"
 }
}
```

γ.. 7  
— 11 — 11

Used properties:

- `word-space (0.6)`

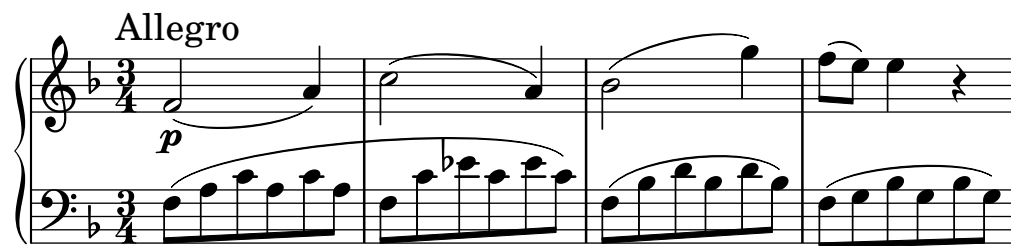


- `multi-measure-rest-number (#t)`
- `multi-measure-rest (#f)`
- `style ('())`

`\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 affetti 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)

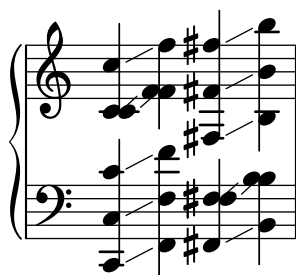
`\stdBassVI` *name* generates a standard bass accordion register symbol for six reed basses.

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 Secció “font-interface” in *Referència de funcionament intern*, Secció “text-interface” in *Referència de funcionament intern* and Secció “instrument-specific-markup-interface” in *Referència de funcionament intern*.

```
\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-from` *arg1* (markup) *arg2* (markup)

Print *arg2* with the dimensions of *arg1*.

`\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 Text markup list commands

The following commands can all be used with `\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` *column-align* (number list) *lst* (markup list)

Returns a table.

*column-align* specifies how each column is aligned, possible values are -1, 0, 1. The number of elements in *column-align* determines how many columns will be printed. The entries to print are given by *lst*, a markup-list. If needed, the last row is filled up with `point-stencils`. Overriding `padding` may be used to increase columns horizontal distance. Overriding `baseline-skip` to increase rows vertical distance.

```
\markuplist {
 \override #'(padding . 2)
 \table
 #'(0 1 0 -1)
 {
 \underline { center-aligned right-aligned center-aligned left-aligned }
 one
 umber 1 thousandth
 umber 0.001
 eleven
 umber 11 hundredth
 umber 0.01
 twenty
 umber 20 tenth
 umber 0.1
 thousand
 umber 1000 one
 umber 1.0
 }
}
```

center-aligned   right-aligned   center-aligned   left-aligned

|        |           |          |            |
|--------|-----------|----------|------------|
| one    | umber     | 1        | thousandth |
| umber  | 0.001     | eleven   | umber      |
| 11     | hundredth | umber    | 0.01       |
| twenty | umber     | 20       | tenth      |
| umber  | 0.1       | thousand | umber      |
| 1000   | one       | umber    | 1.0        |

Used properties:

- `baseline-skip`

- padding (0)

`\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 List of special characters

The following special characters references can be used; for more details, see [ASCII aliases], pàgina 504.

The HTML syntax is used and most of these references are the same as HTML. The rest of them are inspired by L<sup>A</sup>T<sub>E</sub>X.

The characters are boxed so that you can see their size. A small padding has been added between the character and the box for more readability.

|                           |                                                                                     |                            |                                                                                     |                          |                                                                                     |                          |                                                                                       |
|---------------------------|-------------------------------------------------------------------------------------|----------------------------|-------------------------------------------------------------------------------------|--------------------------|-------------------------------------------------------------------------------------|--------------------------|---------------------------------------------------------------------------------------|
| <code>&amp;hellip;</code> |  | <code>&amp;ndash;</code>   |  | <code>&amp;mdash;</code> |  | <code>&amp;iexcl;</code> |  |
| <code>&amp;iquest;</code> |  | <code>&amp;solidus;</code> |  | <code>&amp;flq;</code>   |  | <code>&amp;frq;</code>   |  |
| <code>&amp;flqq;</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>   |  |

|              |     |             |     |            |     |              |    |
|--------------|-----|-------------|-----|------------|-----|--------------|----|
| &zwnj;       | ◌   | &middot;    | ▪   | &bull;     | ◼   | &copyright;  | ©  |
| &registered; | ®   | &trademark; | ™   | &dagger;   | †   | &Dagger;     | ‡  |
| &numero;     | №   | &ordf;      | ª   | &ordm;     | º   | &para;       | ¶  |
| &sect;       | §   | &deg;       | °   | &numero;   | №   | &permil;     | ‰  |
| &brvbar;     | ⏚   | &acute;     | ◌́  | &acutedbl; | ◌̂  | &grave;      | ◌̀ |
| &breve;      | ◌̆  | &caron;     | ◌̈́ | &cedilla;  | ◌̈́ | &circumflex; | ◌̂ |
| &diaeresis;  | ◌̈́ | &macron;    | ◌̄  | &aa;       | Å   | &AA;         | Å  |
| &ae;         | æ   | &AE;        | Æ   | &auml;     | ä   | &Auml;       | Ä  |
| &dh;         | ð   | &DH;        | Ð   | &dj;       | đ   | &DJ;         | Đ  |
| &l;          | ł   | &L;         | Ł   | &ng;       | ŋ   | &NG;         | Ŋ  |
| &o;          | ø   | &O;         | Ø   | &oe;       | œ   | &OE;         | Œ  |
| &ouml;       | ö   | &Ouml;      | Ö   | &s;        | ſ   | &ss;         | ß  |
| &th;         | þ   | &TH;        | Þ   | &uuml;     | ü   | &Uuml;       | Ü  |
| &plus;       | +   | &minus;     | =   | &times;    | ×   | &div;        | ÷  |
| &sup1;       | ¹   | &sup2;      | ²   | &sup3;     | ³   | &sqrt;       | √  |
| &increment;  | Δ   | &infty;     | ∞   | &sum;      | Σ   | &pm;         | ±  |
| &bulletop;   | ◼   | &partial;   | ∂   | &neg;      | ¬   | &currency;   | ¤  |
| &dollar;     | \$  | &euro;      | €   | &pounds;   | £   | &yen;        | ¥  |
| &cent;       | ¢   |             |     |            |     |              |    |

## A.14 List of articulations

The following lists show all the scripts in the Feta font that may be attached to notes (eg. ‘`\f\accent`’ or ‘`\f->`’). Each example shows the script in the *up*, *down* and *neutral* positions respectively.

### Articulation scripts

`\accent` or `->`



`\espressivo`



`\marcato` or `-^`



`\portato` or `-_`



`\staccatissimo`  
or `-!`



`\staccato` or `-.`



`\tenuto` or `--`



### Ornament scripts

`\prall`



`\prallup`



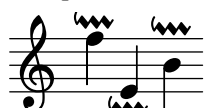
`\pralldown`



`\upprall`



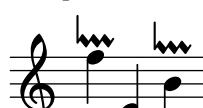
`\downprall`



`\prallprall`



`\lineprall`



`\prallmordent`



`\mordent`



`\upmordent`



`\downmordent`



`\trill`



`\turn`



`\reverseturn`



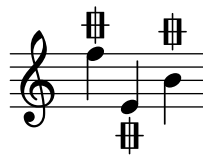
### Fermata scripts

`\shortfermata``\fermata``\longfermata``\verylongfermata`

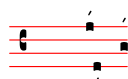
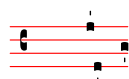
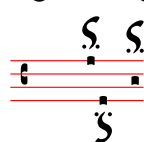
## Instrument-specific scripts

`\upbow``\downbow``\flageolet``\open``\halfopen``\lheel``\rheel``\ltoe``\rtoe``\snappizzicato``\stopped or -+`

## Repeat sign scripts

`\segno``\coda``\varcoda`

## Ancient scripts

`\accentus``\circulus``\ictus``\semicirculus``\signumcongruentiae`

## A.15 Percussion notes

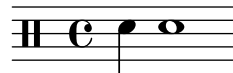
bassdrum  
bd



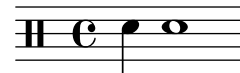
acousticbassdrum  
bda



snare  
sn



acousticsnare  
sna



electricsnare  
sne



lowfloortom  
tomfl



highfloortom  
tomfh



lowtom  
toml



hightom  
tomh



lowmidtom  
tomml



highmidtom  
tommh



highhat  
hh



closedhihat  
hhc



openhighhat  
hho



halfopenhihat  
hhho



pedalhihat  
hhp



crashcymbal  
cymc



crashcymbala  
cymca



crashcymbalb  
cymcb



ridecymbal  
cymr



ridecymbala  
cymra



ridecymbalb  
cymrb



chinese cymbal  
cymch



splashcymbal  
cyms



ridebell  
rb



cowbell  
cb



hibongo  
boh



openhibongo  
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 Technical glossary

A glossary of the technical terms and concepts used internally in LilyPond. These terms may appear in the manuals, on mailing lists or in the source code.

### alist

An association list or **alist** for short is a Scheme pair which associates a value with a key: (key . value). For example, in `scm/lily.scm`, the alist “type-p-name-alist” associates certain type predicates (e.g., `ly:music?`) with names (e.g., “music”) so that type-check failures can be reported with a console message that includes the name of the expected type predicate.

### callback

A **callback** is a routine, function or method whose reference is passed as an argument in a call to another routine, so allowing the called routine to invoke it. The technique enables a lower-level software layer to call a function defined in a higher layer. Callbacks are used extensively in LilyPond to permit user-level Scheme code to define how many low-level actions are performed.

### closure

In Scheme, a **closure** is created when a function, usually a lambda expression, is passed as a variable. The closure contains the function’s code plus references to the lexical bindings of the function’s free variables (i.e., those variables used in the expression but defined outside it). When this function is applied to different arguments later, the free variable bindings that were captured in the closure are used to obtain the values of the free variables to be used in the calculation. One useful property of closures is the retention of internal variable values between invocations, so permitting state to be maintained.

## glyph

A **glyph** is a particular graphical representation of a typographic character, or a combination of two characters forming a ligature. A set of glyphs with a single style and shape comprise a font, and a set of fonts covering several styles and sizes comprise a typeface.

## Vegeu també

Notation Reference: Secció 1.8.3 [Fonts], pàgina 251, Secció 3.3.3 [Special characters], pàgina 502.

## grob

LilyPond objects which represent items of notation in the printed output such as note heads, stems, slurs, ties, fingering, clefs, etc are called ‘Layout objects’, often known as ‘GRaphical Objects’, or **grobs** for short. They are represented by instances of the **Grob** class.

## Vegeu també

Learning Manual: Secció “Objects and interfaces” in *Manual d’aprenentatge*, Secció “Naming conventions of objects and properties” in *Manual d’aprenentatge*, Secció “Properties of layout objects” in *Manual d’aprenentatge*.

Internals Reference: Secció “grob-interface” in *Referència de funcionament intern*, Secció “All layout objects” in *Referència de funcionament intern*.

## immutable

An **immutable** object is one whose state cannot be modified after creation, in contrast to a mutable object, which can be modified after creation.

In LilyPond, immutable or shared properties define the default style and behavior of grobs. They are shared between many objects. In apparent contradiction to the name, they can be changed using `\override` and `\revert`.

## Vegeu també

Notation Reference: [mutable], pàgina 738.

## interface

Actions and properties which are common to a number of grobs are grouped together in an object called a **grob-interface**, or just ‘interface’ for short.

## Vegeu també

Learning Manual: Secció “Objects and interfaces” in *Manual d’aprenentatge*, Secció “Naming conventions of objects and properties” in *Manual d’aprenentatge*, Secció “Properties found in interfaces” in *Manual d’aprenentatge*.

Notation Reference: Secció 5.2.2 [Layout interfaces], pàgina 591.

Internals Reference: Secció “Graphical Object Interfaces” in *Referència de funcionament intern*.

## 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], pàgina 738. 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

A **mutable** object is one whose state can be modified after creation, in contrast to an immutable object, whose state is fixed at the time of creation.

In LilyPond, mutable properties contain values that are specific to one grob. Typically, lists of other objects or results from computations are stored in mutable properties.

## Vegeu també

Notation Reference: [immutable], pàgina 737.

## output-def

An instance of the **Output-def** class contains the methods and data structures associated with an output block. Instances are created for midi, layout and paper blocks.

## parser

A **parser** analyzes the sequence of tokens produced by a lexer to determine its grammatical structure, grouping the tokens progressively into larger groupings according to the rules of the grammar. If the sequence of tokens is valid the end product is a tree of tokens whose root is the grammar's start symbol. If this cannot be achieved the file is invalid and an appropriate error message is produced. The syntactic groupings and the rules for constructing the groupings from their parts for the LilyPond syntax are defined in `lily/parser.yy` and shown in Backus Normal Form (BNF) in Secció “LilyPond grammar” in *Guia del col·laborador*. This file is used to build the parser during the program build by the parser generator, Bison. It is part of the source code and is not included in the LilyPond binary installation.

## parser variable

These are variables defined directly in Scheme. Their direct use by users is strongly discouraged, because their scoping semantics can be confusing.

When the value of such a variable is changed in a `.ly` file, the change is global, and unless explicitly reverted, the new value will persist to the end of the file, affecting subsequent `\score` blocks as well as external files added with the `\include` command. This can lead to unintended consequences and in complex typesetting projects the consequent errors can be difficult to track down.

LilyPond uses the following parser variables:

- `afterGraceFraction`
- `musicQuotes`
- `mode`
- `output-count`
- `output-suffix`
- `partCombineListener`
- `pitchnames`
- `toplevel-bookparts`
- `toplevel-scores`
- `showLastLength`
- `showFirstLength`

## prob

Property Objects, or **probs** for short, are instances of the **Prob** class, a simple base class for objects which have mutable and immutable property alists and the methods to manipulate

them. The `Music` and `Stream_event` classes derive from `Prob`. Instances of the `Prob` class are also created to hold the formatted content of system grobs and titling blocks during page layout.

## smob

**Smobs**, or ScheMe OBjects, are part of the mechanism used by Guile to export C and C++ objects to Scheme code. In LilyPond, smobs are created from C++ objects through macros. There are two types of smob objects: simple smobs, intended for simple immutable objects like numbers, and complex smobs, used for objects with identities. If you have access to the LilyPond sources, more information can be found in `lily/includes/smob.hh`.

## stencil

An instance of the **stencil** class holds the information required to print a typographical object. It is a simple smob containing a confining box, which defines the vertical and horizontal extents of the object, and a Scheme expression which will print the object when evaluated. Stencils may be combined to form more complex stencils defined by a tree of Scheme expressions formed from the Scheme expressions of the component stencils.

The **stencil** property, which connects a grob to its stencil, is defined in the **grob-interface** interface.

## Vegeu també

Internals Reference: Secció “grob-interface” in *Referència de funcionament intern*.

## A.17 All context properties

**accidentalGrouping** (symbol)

If set to 'voice, accidentals on the same note in different octaves may be horizontally staggered if in different voices.

**additionalBassStrings** (list)

The additional tablature bass-strings, which will not get a separate line in `TabStaff`. It is a list of the pitches of each string (starting with the lowest numbered one).

**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 Secció “Score” in *Referència de funcionament intern* then all staves share accidentals, and if *context* is Secció “Staff” in *Referència de funcionament intern* 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 Secció “Timing\_translator” in *Referència de funcionament intern* at Secció “Score” in *Referència de funcionament intern* 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 Secció "Lyrics" in *Referència de funcionament intern* 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 Secció "Lyrics" in *Referència de funcionament intern*.

**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.

`soloIIText` (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 Secció “Pitch_squash-engraver” in *Referència de funcionament intern*.

`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 Layout properties

`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 Secció “break-alignment-interface” in *Referència de funcionament intern*).

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 Secció “break-alignment-interface” in *Referència de funcionament intern*.

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 Secció “break-alignment-interface” in *Referència de funcionament intern*.

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 Secció “note-collision-interface” in *Referència de funcionament intern*.

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 $\text{thickness} * (1 + \text{string-thickness-factor})^{(k-1)}$. 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 Secció “note-collision-interface” in *Referència de funcionament intern*.

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)

Vertical positions of ledger lines. When set on a **StaffSymbol** grob it defines a repeating pattern of ledger lines and any parenthesized groups will always be shown together.

ledger-positions-function (any type)

A quoted Scheme procedure that takes a **StaffSymbol** grob and the vertical position of a note head as arguments and returns a list of ledger line positions.

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 Secció “Formatting text” in *Referència de la notació*.

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 Secció “note-collision-interface” in *Referència de funcionament intern*.
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 and keys.

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 it is 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** (index or symbol)
When set as a positive 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 independent of the **Keep_alive_together_engraver**. Set to '(), the layer does not participate in the layering decisions. The property can also be set as a symbol for common behaviors: **#'any** to keep the layer alive with any other layer in the group; **#'above** or **#'below** to keep the layer alive with the context immediately before or after it, respectively.
- 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 Secció “spacing-spanner-interface” in *Referència de funcionament intern*.

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 Secció “break-alignment-interface” in *Referència de funcionament intern*. 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 Secció “spacing-spanner-interface” in *Referència de funcionament intern*.

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 (index or symbol)

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 Secció "Formatting text" in *Referència de la notació*.

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 Available music functions

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] - *fraction* [fraction, as pair] *main* (music) *grace* (music)

Create *grace* note(s) after a *main* music expression.

The musical position of the grace expression is after a given fraction of the main note’s duration has passed. If *fraction* is not specified as first argument, it is taken from **afterGraceFraction** which has a default value of 3/4.

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* (key list or symbol) *arg* (list) *item* (key 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* (key 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* (list of indexes or symbols) *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* (list of indexes or symbols) *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* (list of indexes or symbols)

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* (key list or symbol) *value* (any type) *item* (key 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* (key 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* (key 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* (index or symbol) *event* (*post event*)

This sets the `spanner-id` property of the following *event* to the given *id* (non-negative integer or symbol). 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 Context modification identifiers

The following commands are defined for use as context modifications within a `\layout` or `\with` block.

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 Secció ‘‘VerticalAxisGroup’’ in *Referència de funcionament intern* to `#t`.
- Sets grob property `remove-first` in Secció ‘‘VerticalAxisGroup’’ in *Referència de funcionament intern* 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 Secció ‘‘VerticalAxisGroup’’ in *Referència de funcionament intern* to `#t`.

A.21 Predefined type predicates

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

odd?	odd number
output-port?	output port
positive?	positive number
rational?	rational number
real?	real number
zero?	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
key?	index or symbol
key-list?	list of indexes or symbols
key-list-or-music?	key list or music
key-list-or-symbol?	key list or symbol
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

<code>ly:box?</code>	box
<code>ly:context?</code>	context
<code>ly:context-def?</code>	context definition
<code>ly:context-mod?</code>	context modification
<code>ly:dimension?</code>	dimension, in staff space
<code>ly:dir?</code>	direction
<code>ly:dispatcher?</code>	dispatcher
<code>ly:duration?</code>	duration
<code>ly:event?</code>	post event
<code>ly:font-metric?</code>	font metric
<code>ly:grob?</code>	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 Scheme functions

<code>ly:add-context-mod</code>	<i>contextmods</i> <i>modification</i>	[Funció]
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>	[Funció]
Add mappings for error messages from <i>alist</i> .		

ly:add-interface <i>iface desc props</i>	[Funció]
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.	
ly:add-listener <i>callback disp cl</i>	[Funció]
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.	
ly:add-option <i>sym val description</i>	[Funció]
Add a program option <i>sym</i> . <i>val</i> is the default value and <i>description</i> is a string description.	
ly:all-grob-interfaces	[Funció]
Return the hash table with all grob interface descriptions.	
ly:all-options	[Funció]
Get all option settings in an alist.	
ly:all-stencil-expressions	[Funció]
Return all symbols recognized as stencil expressions.	
ly:angle <i>x y</i>	[Funció]
Calculates angle in degrees of given vector. With one argument, <i>x</i> is a number pair indicating the vector. With two arguments, <i>x</i> and <i>y</i> specify the respective coordinates.	
ly:assoc-get <i>key alist default-value strict-checking</i>	[Funció]
Return value if <i>key</i> in <i>alist</i> , else <i>default-value</i> (or #f if not specified). If <i>strict-checking</i> is set to #t and <i>key</i> is not in <i>alist</i> , a programming_error is output.	
ly:axis-group-interface::add-element <i>grob grob-element</i>	[Funció]
Set <i>grob</i> the parent of <i>grob-element</i> on all axes of <i>grob</i> .	
ly:basic-progress <i>str rest</i>	[Funció]
A Scheme callable function to issue a basic progress message <i>str</i> . The message is formatted with format and <i>rest</i> .	
ly:beam-score-count	[Funció]
count number of beam scores.	
ly:bigpdfs	[Funció]
Return true if the command line includes the --bigpdf parameter.	
ly:book? <i>x</i>	[Funció]
Is <i>x</i> a Book object?	
ly:book-add-bookpart! <i>book-smob book-part</i>	[Funció]
Add <i>book-part</i> to <i>book-smob</i> book part list.	
ly:book-add-score! <i>book-smob score</i>	[Funció]
Add <i>score</i> to <i>book-smob</i> score list.	
ly:book-book-parts <i>book</i>	[Funció]
Return book parts in <i>book</i> .	
ly:book-header <i>book</i>	[Funció]
Return header in <i>book</i> .	
ly:book-paper <i>book</i>	[Funció]
Return paper in <i>book</i> .	

ly:book-process <i>book-smob default-paper default-layout output</i>	[Funció]
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).	
ly:book-process-to-systems <i>book-smob default-paper default-layout output</i>	[Funció]
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).	
ly:book-scores <i>book</i>	[Funció]
Return scores in <i>book</i> .	
ly:book-set-header! <i>book module</i>	[Funció]
Set the book header.	
ly:box? <i>x</i>	[Funció]
Is <i>x</i> a Box object?	
ly:bp <i>num</i>	[Funció]
<i>num</i> bigpoints (1/72th inch).	
ly:bracket <i>a iv t p</i>	[Funció]
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> .	
ly:broadcast <i>disp ev</i>	[Funció]
Send the stream event <i>ev</i> to the dispatcher <i>disp</i> .	
ly:camel-case->lisp-identifier <i>name-sym</i>	[Funció]
Convert FooBar_Bla to foo-bar-bla style symbol.	
ly:chain-assoc-get <i>key achain default-value strict-checking</i>	[Funció]
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 <i>#f</i> if <i>default-value</i> is not specified. With <i>strict-checking</i> set to <i>#t</i> , a programming_error is output in such cases.	
ly:check-expected-warnings	[Funció]
Check whether all expected warnings have really been triggered.	
ly:cm <i>num</i>	[Funció]
<i>num</i> cm.	
ly:command-line-code	[Funció]
The Scheme code specified on command-line with <i>-e</i> .	
ly:command-line-options	[Funció]
The Scheme options specified on command-line with <i>-d</i> .	
ly:connect-dispatchers <i>to from</i>	[Funció]
Make the dispatcher <i>to</i> listen to events from <i>from</i> .	
ly:context? <i>x</i>	[Funció]
Is <i>x</i> a Context object?	
ly:context-current-moment <i>context</i>	[Funció]
Return the current moment of <i>context</i> .	

<code>ly:context-def? x</code>	[Funció]
Is <i>x</i> a <code>Context_def</code> object?	
<code>ly:context-def-lookup def sym val</code>	[Funció]
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>	[Funció]
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>	[Funció]
Return <code>event-source</code> of context <i>context</i> .	
<code>ly:context-events-below context</code>	[Funció]
Return a <code>stream-distributor</code> that distributes all events from <i>context</i> and all its subcontexts.	
<code>ly:context-find context name</code>	[Funció]
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 context name</code>	[Funció]
Return the definition of <i>name</i> (a symbol) within <i>context</i> as an alist.	
<code>ly:context-id context</code>	[Funció]
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 context grob cell</code>	[Funció]
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? x</code>	[Funció]
Is <i>x</i> a <code>Context_mod</code> object?	
<code>ly:context-mod-apply! context mod</code>	[Funció]
Apply the context modification <i>mod</i> to <i>context</i> .	
<code>ly:context-name context</code>	[Funció]
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 context</code>	[Funció]
Return <code>now-moment</code> of context <i>context</i> .	
<code>ly:context-parent context</code>	[Funció]
Return the parent of <i>context</i> , <code>#f</code> if none.	
<code>ly:context-property context sym def</code>	[Funció]
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 context name</code>	[Funció]
Return the context above <i>context</i> where <i>name</i> is defined.	
<code>ly:context-pushpop-property context grob eltprop val</code>	[Funció]
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).	

ly:context-set-property! <i>context name val</i>	[Funció]
Set value of property <i>name</i> in context <i>context</i> to <i>val</i> .	
ly:context-unset-property <i>context name</i>	[Funció]
Unset value of property <i>name</i> in context <i>context</i> .	
ly:debug <i>str rest</i>	[Funció]
A Scheme callable function to issue a debug message <i>str</i> . The message is formatted with <i>format</i> and <i>rest</i> .	
ly:default-scale	[Funció]
Get the global default scale.	
ly:dimension? <i>d</i>	[Funció]
Return <i>d</i> as a number. Used to distinguish length variables from normal numbers.	
ly:dir? <i>s</i>	[Funció]
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.	
ly:directed <i>direction magnitude</i>	[Funció]
Calculates an (<i>x</i> . <i>y</i>) pair with optional <i>magnitude</i> (defaulting to 1.0) and <i>direction</i> specified either as an angle in degrees or a coordinate pair giving the direction. If <i>magnitude</i> is a pair, the respective coordinates are scaled independently, useful for ellipse drawings.	
ly:disconnect-dispatchers <i>to from</i>	[Funció]
Stop the dispatcher <i>to</i> listening to events from <i>from</i> .	
ly:dispatcher? <i>x</i>	[Funció]
Is <i>x</i> a Dispatcher object?	
ly:duration? <i>x</i>	[Funció]
Is <i>x</i> a Duration object?	
ly:duration<? <i>p1 p2</i>	[Funció]
Is <i>p1</i> shorter than <i>p2</i> ?	
ly:duration->string <i>dur</i>	[Funció]
Convert <i>dur</i> to a string.	
ly:duration-dot-count <i>dur</i>	[Funció]
Extract the dot count from <i>dur</i> .	
ly:duration-factor <i>dur</i>	[Funció]
Extract the compression factor from <i>dur</i> . Return it as a pair.	
ly:duration-length <i>dur</i>	[Funció]
The length of the duration as a moment .	
ly:duration-log <i>dur</i>	[Funció]
Extract the duration log from <i>dur</i> .	
ly:duration-scale <i>dur</i>	[Funció]
Extract the compression factor from <i>dur</i> . Return it as a rational.	
ly:effective-prefix	[Funció]
Return effective prefix.	

- ly:encode-string-for-pdf** *str* [Funció]
 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).
- ly:engraver-announce-end-grob** *engraver grob cause* [Funció]
 Announce the end of a grob (i.e., the end of a spanner) originating from given *engraver* instance, with *grob* being a grob. *cause* should either be another grob or a music event.
- ly:engraver-make-grob** *engraver grob-name cause* [Funció]
 Create a grob originating from given *engraver* instance, with given *grob-name*, a symbol. *cause* should either be another grob or a music event.
- ly:error** *str rest* [Funció]
 A Scheme callable function to issue the error *str*. The error is formatted with **format** and *rest*.
- ly:event?** *obj* [Funció]
 Is *obj* a proper (non-rhythmic) event object?
- ly:event-deep-copy** *m* [Funció]
 Copy *m* and all sub expressions of *m*.
- ly:event-property** *sev sym val* [Funció]
 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* [Funció]
 Set property *sym* in event *ev* to *val*.
- ly:expand-environment** *str* [Funció]
 Expand **\$VAR** and **\${VAR}** in *str*.
- ly:expect-warning** *str rest* [Funció]
 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* [Funció]
 Return the absolute file name of *name*, or **#f** if not found.
- ly:font-config-add-directory** *dir* [Funció]
 Add directory *dir* to FontConfig.
- ly:font-config-add-font** *font* [Funció]
 Add font *font* to FontConfig.
- ly:font-config-display-fonts** [Funció]
 Dump a list of all fonts visible to FontConfig.
- ly:font-config-get-font-file** *name* [Funció]
 Get the file for font *name*.
- ly:font-design-size** *font* [Funció]
 Given the font metric *font*, return the design size, relative to the current output-scale.
- ly:font-file-name** *font* [Funció]
 Given the font metric *font*, return the corresponding file name.

ly:font-get-glyph *font name* [Funció]

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-charcode *font name* [Funció]

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* [Funció]

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* [Funció]

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* [Funció]

Given the font metric *font*, return the magnification, relative to the current output-scale.

ly:font-metric? *x* [Funció]

Is *x* a **Font_metric** object?

ly:font-name *font* [Funció]

Given the font metric *font*, return the corresponding name.

ly:font-sub-fonts *font* [Funció]

Given the font metric *font* of an OpenType font, return the names of the subfonts within *font*.

ly:format *str rest* [Funció]

LilyPond specific format, supporting **~a** and **~[0-9]f**. Basic support for **~s** is also provided.

ly:format-output *context* [Funció]

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* [Funció]

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 [Funció]

Get a hash table with all LilyPond Scheme extension functions.

ly:get-all-translators [Funció]

Return a list of all translator objects that may be instantiated.

- ly:get-cff-offset** *font-file-name idx* [Funció]
 Get the offset of 'CFF' table for *font-file-name*, returning it as an integer. The optional *idx* argument is useful for OpenType/CFF collections (OTC) only; it specifies the font index within the OTC. The default value of *idx* is 0.
- ly:get-context-mods** *contextmod* [Funció]
 Returns the list of context modifications stored in *contextmod*.
- ly:get-font-format** *font-file-name idx* [Funció]
 Get the font format for *font-file-name*, returning it as a symbol. The optional *idx* argument is useful for TrueType Collections (TTC) and OpenType/CFF collections (OTC) only; it specifies the font index within the TTC/OTC. The default value of *idx* is 0.
- ly:get-option** *var* [Funció]
 Get a global option setting.
- ly:get-spacing-spec** *from-scm to-scm* [Funció]
 Return the spacing spec going between the two given grobs, *from-scm* and *to-scm*.
- ly:get-undead** *undead* [Funció]
 Get back object from *undead*.
- ly:gettext** *original* [Funció]
 A Scheme wrapper function for **gettext**.
- ly:grob?** *x* [Funció]
 Is *x* a Grob object?
- ly:grob-alist-chain** *grob global* [Funció]
 Get an alist chain for grob *grob*, with *global* as the global default. If unspecified, **font-defaults** from the layout block is taken.
- ly:grob-array?** *x* [Funció]
 Is *x* a Grob_array object?
- ly:grob-array->list** *grob-arr* [Funció]
 Return the elements of *grob-arr* as a Scheme list.
- ly:grob-array-length** *grob-arr* [Funció]
 Return the length of *grob-arr*.
- ly:grob-array-ref** *grob-arr index* [Funció]
 Retrieve the *index*th element of *grob-arr*.
- ly:grob-basic-properties** *grob* [Funció]
 Get the immutable properties of *grob*.
- ly:grob-chain-callback** *grob proc sym* [Funció]
 Find the callback that is stored as property *sym* of grob *grob* and chain *proc* to the head of this, meaning that it is called using *grob* and the previous callback's result.
- ly:grob-common-refpoint** *grob other axis* [Funció]
 Find the common refpoint of *grob* and *other* for *axis*.
- ly:grob-common-refpoint-of-array** *grob others axis* [Funció]
 Find the common refpoint of *grob* and *others* (a grob-array) for *axis*.

ly:grob-default-font <i>grob</i>	[Funció]
Return the default font for grob <i>grob</i> .	
ly:grob-extent <i>grob refp axis</i>	[Funció]
Get the extent in <i>axis</i> direction of <i>grob</i> relative to the grob <i>refp</i> .	
ly:grob-get-vertical-axis-group-index <i>grob</i>	[Funció]
Get the index of the vertical axis group the grob <i>grob</i> belongs to; return -1 if none is found.	
ly:grob-interfaces <i>grob</i>	[Funció]
Return the interfaces list of grob <i>grob</i> .	
ly:grob-layout <i>grob</i>	[Funció]
Get \layout definition from grob <i>grob</i> .	
ly:grob-object <i>grob sym</i>	[Funció]
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> .	
ly:grob-original <i>grob</i>	[Funció]
Return the unbroken original grob of <i>grob</i> .	
ly:grob-parent <i>grob axis</i>	[Funció]
Get the parent of <i>grob</i> . <i>axis</i> is 0 for the X-axis, 1 for the Y-axis.	
ly:grob-pq<? <i>a b</i>	[Funció]
Compare two grob priority queue entries. This is an internal function.	
ly:grob-properties <i>grob</i>	[Funció]
Get the mutable properties of <i>grob</i> .	
ly:grob-properties? <i>x</i>	[Funció]
Is <i>x</i> a <code>Grob_properties</code> object?	
ly:grob-property <i>grob sym val</i>	[Funció]
Return the value for property <i>sym</i> of <i>grob</i> . If no value is found, return <i>val</i> or '()' if <i>val</i> is not specified.	
ly:grob-property-data <i>grob sym</i>	[Funció]
Return the value for property <i>sym</i> of <i>grob</i> , but do not process callbacks.	
ly:grob-pure-height <i>grob refp beg end val</i>	[Funció]
Return the pure height of <i>grob</i> given retpoint <i>refp</i> . If no value is found, return <i>val</i> or '()' if <i>val</i> is not specified.	
ly:grob-pure-property <i>grob sym beg end val</i>	[Funció]
Return the pure value for property <i>sym</i> of <i>grob</i> . If no value is found, return <i>val</i> or '()' if <i>val</i> is not specified.	
ly:grob-relative-coordinate <i>grob refp axis</i>	[Funció]
Get the coordinate in <i>axis</i> direction of <i>grob</i> relative to the grob <i>refp</i> .	
ly:grob-robust-relative-extent <i>grob refp axis</i>	[Funció]
Get the extent in <i>axis</i> direction of <i>grob</i> relative to the grob <i>refp</i> , or (0,0) if empty.	
ly:grob-script-priority-less <i>a b</i>	[Funció]
Compare two grobs by script priority. For internal use.	

ly:grob-set-nested-property! <i>grob symlist val</i>	[Funció]
Set nested property <i>symlist</i> in grob <i>grob</i> to value <i>val</i> .	
ly:grob-set-object! <i>grob sym val</i>	[Funció]
Set <i>sym</i> in grob <i>grob</i> to value <i>val</i> .	
ly:grob-set-parent! <i>grob axis parent-grob</i>	[Funció]
Set <i>parent-grob</i> the parent of grob <i>grob</i> in axis <i>axis</i> .	
ly:grob-set-property! <i>grob sym val</i>	[Funció]
Set <i>sym</i> in grob <i>grob</i> to value <i>val</i> .	
ly:grob-spanned-rank-interval <i>grob</i>	[Funció]
Returns a pair with the rank of the furthest left column and the rank of the furthest right column spanned by <i>grob</i> .	
ly:grob-staff-position <i>sg</i>	[Funció]
Return the Y-position of <i>sg</i> relative to the staff.	
ly:grob-suicide! <i>grob</i>	[Funció]
Kill <i>grob</i> .	
ly:grob-system <i>grob</i>	[Funció]
Return the system grob of <i>grob</i> .	
ly:grob-translate-axis! <i>grob d a</i>	[Funció]
Translate <i>grob</i> on axis <i>a</i> over distance <i>d</i> .	
ly:grob-vertical <? <i>a b</i>	[Funció]
Does <i>a</i> lie above <i>b</i> on the page?	
ly:gulp-file <i>name size</i>	[Funció]
Read <i>size</i> characters from the file <i>name</i> , and return its contents in a string. If <i>size</i> is undefined, the entire file is read. The file is looked up using the search path.	
ly:has-glyph-names? <i>font-file-name idx</i>	[Funció]
Does the font for <i>font_file_name</i> have glyph names? The optional <i>idx</i> argument is useful for TrueType Collections (TTC) and OpenType/CFF collections (OTC) only; it specifies the font index within the TTC/OTC. The default value of <i>idx</i> is 0.	
ly:hash-table-keys <i>tab</i>	[Funció]
Return a list of keys in <i>tab</i> .	
ly:inch <i>num</i>	[Funció]
<i>num</i> inches.	
ly:input-both-locations <i>sip</i>	[Funció]
Return input location in <i>sip</i> as (file-name first-line first-column last-line last-column).	
ly:input-file-line-char-column <i>sip</i>	[Funció]
Return input location in <i>sip</i> as (file-name line char column).	
ly:input-location? <i>x</i>	[Funció]
Is <i>x</i> a Input object?	
ly:input-message <i>sip msg rest</i>	[Funció]
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 format 's argument, using <i>rest</i> .	

- ly:input-warning** *sip msg rest* [Funció]
 Print *msg* as a GNU compliant warning message, pointing to the location in *sip*. *msg* is interpreted similar to **format**'s argument, using *rest*.
- ly:interpret-music-expression** *mus ctx* [Funció]
 Interpret the music expression *mus* in the global context *ctx*. The context is returned in its final state.
- ly:interpret-stencil-expression** *expr func arg1 offset* [Funció]
 Parse *expr*, feed bits to *func* with first arg *arg1* having offset *offset*.
- ly:intlog2** *d* [Funció]
 The 2-logarithm of $1/d$.
- ly:item?** *g* [Funció]
 Is *g* an **Item** object?
- ly:item-break-dir** *it* [Funció]
 The break status direction of item *it*. -1 means end of line, 0 unbroken, and 1 beginning of line.
- ly:item-get-column** *it* [Funció]
 Return the **PaperColumn** or **NonMusicalPaperColumn** associated with this **Item**.
- ly:iterator?** *x* [Funció]
 Is *x* a **Music_iterator** object?
- ly:length** *x y* [Funció]
 Calculates magnitude of given vector. With one argument, *x* is a number pair indicating the vector. With two arguments, *x* and *y* specify the respective coordinates.
- ly:lexer-keywords** *lexer* [Funció]
 Return a list of (KEY . CODE) pairs, signifying the LilyPond reserved words list.
- ly:lily-lexer?** *x* [Funció]
 Is *x* a **Lily_lexer** object?
- ly:lily-parser?** *x* [Funció]
 Is *x* a **Lily_parser** object?
- ly:line-interface::line** *grob startx starty endx endy* [Funció]
 Make a line using layout information from grob *grob*.
- ly:listened-event-class?** *disp cl* [Funció]
 Does *disp* listen to any event type in the list *cl*?
- ly:listened-event-types** *disp* [Funció]
 Return a list of all event types that *disp* listens to.
- ly:listener?** *x* [Funció]
 Is *x* a **Listener** object?
- ly:make-book** *paper header scores* [Funció]
 Make a \book of *paper* and *header* (which may be **#f** as well) containing \scores.
- ly:make-book-part** *scores* [Funció]
 Make a \bookpart containing \scores.

- ly:make-context-mod** *mod-list* [Funció]
Creates a context modification, optionally initialized via the list of modifications *mod-list*.
- ly:make-dispatcher** [Funció]
Return a newly created dispatcher.
- ly:make-duration** *length dotcount num den* [Funció]
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* [Funció]
Set up a global interpretation context, using the output block *output-def*. The context is returned.
- ly:make-global-translator** *global* [Funció]
Create a translator group and connect it to the global context *global*. The translator group is returned.
- ly:make-grob-properties** *alist* [Funció]
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* [Funció]
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* [Funció]
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* [Funció]
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* [Funció]
Make *music* relative to *pitch*, return final pitch.
- ly:make-output-def** [Funció]
Make an output definition.
- ly:make-page-label-marker** *label* [Funció]
Return page marker with label *label*.

- ly:make-page-permission-marker** *symbol permission* [Funció]
Return page marker with page breaking and turning permissions.
- ly:make-pango-description-string** *chain size* [Funció]
Make a `PangoFontDescription` string for the property alist *chain* at size *size*.
- ly:make-paper-outputter** *port format* [Funció]
Create an outputter that evaluates within *output-format*, writing to *port*.
- ly:make-pitch** *octave note alter* [Funció]
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* [Funció]
Create a `Prob` object.
- ly:make-scale** *steps* [Funció]
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* [Funció]
Return score with *music* encapsulated in it.
- ly:make-spring** *ideal min-dist* [Funció]
Make a spring. *ideal* is the ideal distance of the spring, and *min-dist* is the minimum distance.
- ly:make-stencil** *expr xext yext* [Funció]
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* [Funció]
Create a stream event of class *cl* with the given mutable property list.
- ly:make-undead** *object* [Funció]
This packages *object* in a manner that keeps it from triggering "Parsed object should be dead" messages.
- ly:make-unpure-pure-container** *unpure pure* [Funció]
Make an unpure-pure container. *unpure* should be an unpure expression, and *pure* should be a pure expression. If *pure* is omitted, the value of *unpure* will be used twice, except that a callback is given two extra arguments that are ignored for the sake of pure calculations.
- ly:message** *str rest* [Funció]
A Scheme callable function to issue the message *str*. The message is formatted with `format` and *rest*.
- ly:minimal-breaking** *pb* [Funció]
Break (pages and lines) the `Paper_book` object *pb* without looking for optimal spacing: stack as many lines on a page before moving to the next one.
- ly:mm** *num* [Funció]
num mm.

<code>ly:module->alist <i>mod</i></code>	[Funció]
Dump the contents of module <i>mod</i> as an alist.	
<code>ly:module-copy <i>dest src</i></code>	[Funció]
Copy all bindings from module <i>src</i> into <i>dest</i> .	
<code>ly:modules-lookup <i>modules sym def</i></code>	[Funció]
Look up <i>sym</i> in the list <i>modules</i> , returning the first occurrence. If not found, return <i>def</i> or <code>#f</code> if <i>def</i> isn't specified.	
<code>ly:moment? <i>x</i></code>	[Funció]
Is <i>x</i> a <code>Moment</code> object?	
<code>ly:moment<? <i>a b</i></code>	[Funció]
Compare two moments.	
<code>ly:moment-add <i>a b</i></code>	[Funció]
Add two moments.	
<code>ly:moment-div <i>a b</i></code>	[Funció]
Divide two moments.	
<code>ly:moment-grace <i>mom</i></code>	[Funció]
Extract grace timing as a rational number from <i>mom</i> .	
<code>ly:moment-grace-denominator <i>mom</i></code>	[Funció]
Extract denominator from grace timing.	
<code>ly:moment-grace-numerator <i>mom</i></code>	[Funció]
Extract numerator from grace timing.	
<code>ly:moment-main <i>mom</i></code>	[Funció]
Extract main timing as a rational number from <i>mom</i> .	
<code>ly:moment-main-denominator <i>mom</i></code>	[Funció]
Extract denominator from main timing.	
<code>ly:moment-main-numerator <i>mom</i></code>	[Funció]
Extract numerator from main timing.	
<code>ly:moment-mod <i>a b</i></code>	[Funció]
Modulo of two moments.	
<code>ly:moment-mul <i>a b</i></code>	[Funció]
Multiply two moments.	
<code>ly:moment-sub <i>a b</i></code>	[Funció]
Subtract two moments.	
<code>ly:music? <i>obj</i></code>	[Funció]
Is <i>obj</i> a music object?	
<code>ly:music-compress <i>m factor</i></code>	[Funció]
Compress music object <i>m</i> by moment <i>factor</i> .	
<code>ly:music-deep-copy <i>m origin</i></code>	[Funció]
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.	

ly:music-duration-compress <i>mus fact</i>	[Funció]
Compress <i>mus</i> by factor <i>fact</i> , which is a Moment .	
ly:music-duration-length <i>mus</i>	[Funció]
Extract the duration field from <i>mus</i> and return the length.	
ly:music-function? <i>x</i>	[Funció]
Is <i>x</i> a Music_function object?	
ly:music-function-extract <i>x</i>	[Funció]
Return the Scheme function inside <i>x</i> .	
ly:music-function-signature <i>x</i>	[Funció]
Return the function signature inside <i>x</i> .	
ly:music-length <i>mus</i>	[Funció]
Get the length of music expression <i>mus</i> and return it as a Moment object.	
ly:music-list? <i>lst</i>	[Funció]
Is <i>lst</i> a list of music objects?	
ly:music-mutable-properties <i>mus</i>	[Funció]
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 make-music function.	
ly:music-output? <i>x</i>	[Funció]
Is <i>x</i> a Music_output object?	
ly:music-property <i>mus sym val</i>	[Funció]
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.	
ly:music-set-property! <i>mus sym val</i>	[Funció]
Set property <i>sym</i> in music expression <i>mus</i> to <i>val</i> .	
ly:music-transpose <i>m p</i>	[Funció]
Transpose <i>m</i> such that central C is mapped to <i>p</i> . Return <i>m</i> .	
ly:note-column-accidentals <i>note-column</i>	[Funció]
Return the AccidentalPlacement grob from <i>note-column</i> if any, or SCM_EOL otherwise.	
ly:note-column-dot-column <i>note-column</i>	[Funció]
Return the DotColumn grob from <i>note-column</i> if any, or SCM_EOL otherwise.	
ly:note-head::stem-attachment <i>font-metric glyph-name</i>	[Funció]
Get attachment in <i>font-metric</i> for attaching a stem to notehead <i>glyph-name</i> .	
ly:number->string <i>s</i>	[Funció]
Convert <i>s</i> to a string without generating many decimals.	
ly:one-line-auto-height-breaking <i>pb</i>	[Funció]
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.	
ly:one-line-breaking <i>pb</i>	[Funció]
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.	

- ly:optimal-breaking** *pb* [Funció]
 Optimally break (pages and lines) the `Paper_book` object *pb* to minimize badness in both vertical and horizontal spacing.
- ly:option-usage** *port* [Funció]
 Print `ly:set-option` usage. Optional *port* argument for the destination defaults to current output port.
- ly:otf->cff** *otf-file-name idx* [Funció]
 Convert the contents of an OTF file to a CFF file, returning it as a string. The optional *idx* argument is useful for OpenType/CFF collections (OTC) only; it specifies the font index within the OTC. The default value of *idx* is 0.
- ly:otf-font?** *font* [Funció]
 Is *font* an OpenType font?
- ly:otf-font-glyph-info** *font glyph* [Funció]
 Given the font metric *font* of an OpenType font, return the information about named glyph *glyph* (a string).
- ly:otf-font-table-data** *font tag* [Funció]
 Extract a table *tag* from *font*. Return empty string for non-existent *tag*.
- ly:otf-glyph-count** *font* [Funció]
 Return the number of glyphs in *font*.
- ly:otf-glyph-list** *font* [Funció]
 Return a list of glyph names for *font*.
- ly:output-def?** *x* [Funció]
 Is *x* a `Output_def` object?
- ly:output-def-clone** *def* [Funció]
 Clone output definition *def*.
- ly:output-def-lookup** *def sym val* [Funció]
 Return the value of *sym* in output definition *def* (e.g., `\paper`). If no value is found, return *val* or `()` if *val* is undefined.
- ly:output-def-parent** *def* [Funció]
 Return the parent output definition of *def*.
- ly:output-def-scope** *def* [Funció]
 Return the variable scope inside *def*.
- ly:output-def-set-variable!** *def sym val* [Funció]
 Set an output definition *def* variable *sym* to *val*.
- ly:output-description** *output-def* [Funció]
 Return the description of translators in *output-def*.
- ly:output-find-context-def** *output-def context-name* [Funció]
 Return an alist of all context defs (matching *context-name* if given) in *output-def*.
- ly:output-formats** [Funció]
 Formats passed to `--format` as a list of strings, used for the output.

<code>ly:outputter-close</code> <i>outputter</i>	[Funció]
Close port of <i>outputter</i> .	
<code>ly:outputter-dump-stencil</code> <i>outputter stencil</i>	[Funció]
Dump stencil <i>expr</i> onto <i>outputter</i> .	
<code>ly:outputter-dump-string</code> <i>outputter str</i>	[Funció]
Dump <i>str</i> onto <i>outputter</i> .	
<code>ly:outputter-module</code> <i>outputter</i>	[Funció]
Return output module of <i>outputter</i> .	
<code>ly:outputter-output-scheme</code> <i>outputter expr</i>	[Funció]
Eval <i>expr</i> in module of <i>outputter</i> .	
<code>ly:outputter-port</code> <i>outputter</i>	[Funció]
Return output port for <i>outputter</i> .	
<code>ly:page-marker?</code> <i>x</i>	[Funció]
Is <i>x</i> a <code>Page_marker</code> object?	
<code>ly:page-turn-breaking</code> <i>pb</i>	[Funció]
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>	[Funció]
Is <i>f</i> a pango font?	
<code>ly:pango-font-physical-fonts</code> <i>f</i>	[Funció]
Return alist of (<code>ps-name</code> <code>file-name</code> <code>font-index</code>) lists for Pango font <i>f</i> .	
<code>ly:paper-book?</code> <i>x</i>	[Funció]
Is <i>x</i> a <code>Paper_book</code> object?	
<code>ly:paper-book-header</code> <i>pb</i>	[Funció]
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>	[Funció]
Return pages in <code>Paper_book</code> object <i>pb</i> .	
<code>ly:paper-book-paper</code> <i>pb</i>	[Funció]
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>	[Funció]
Return performances in <code>Paper_book</code> object <i>pb</i> .	
<code>ly:paper-book-scopes</code> <i>pb</i>	[Funció]
Return scopes in <code>Paper_book</code> object <i>pb</i> .	
<code>ly:paper-book-systems</code> <i>pb</i>	[Funció]
Return systems in <code>Paper_book</code> object <i>pb</i> .	
<code>ly:paper-column::print</code>	[Funció]
Optional stencil for <code>PaperColumn</code> or <code>NonMusicalPaperColumn</code> . Draws the rank number 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>	[Funció]
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>	[Funció]
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>	[Funció]
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>	[Funció]
Return the output-scale for output definition <i>def</i> .	
<code>ly:paper-score-paper-systems</code> <i>paper-score</i>	[Funció]
Return vector of <code>paper_system</code> objects from <i>paper-score</i> .	
<code>ly:paper-system?</code> <i>obj</i>	[Funció]
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>	[Funció]
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>	[Funció]
Parse a single <code>.ly</code> file. Upon failure, throw <code>ly-file-failed</code> key.	
<code>ly:parse-string-expression</code> <i>parser-smob ly-code filename line</i>	[Funció]
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>	[Funció]
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>	[Funció]
Clear error flag for <i>parser</i> , defaulting to current parser.	
<code>ly:parser-clone</code> <i>closures location</i>	[Funció]
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>	[Funció]
Bind <i>symbol</i> to <i>val</i> in current parser's module.	
<code>ly:parser-error</code> <i>msg input</i>	[Funció]
Display an error message and make current parser fail. Without a current parser, trigger an ordinary error.	
<code>ly:parser-has-error?</code> <i>parser</i>	[Funció]
Does <i>parser</i> (defaulting to current parser) have an error flag?	
<code>ly:parser-include-string</code> <i>ly-code</i>	[Funció]
Include the string <i>ly-code</i> into the input stream for current parser. Can only be used in immediate Scheme expressions (<code>\$</code> instead of <code>#</code>).	
<code>ly:parser-lexer</code> <i>parser</i>	[Funció]
Return the lexer for <i>parser</i> , defaulting to current parser	

ly:parser-lookup <i>symbol</i>	[Funció]
Look up <i>symbol</i> in current parser's module. Return '()' if not defined.	
ly:parser-output-name <i>parser</i>	[Funció]
Return the base name of the output file. If <i>parser</i> is left off, use currently active parser.	
ly:parser-parse-string <i>parser-smob ly-code</i>	[Funció]
Parse the string <i>ly-code</i> with <i>parser-smob</i> . Upon failure, throw ly-file-failed key.	
ly:parser-set-note-names <i>names</i>	[Funció]
Replace current note names in parser. <i>names</i> is an alist of symbols. This only has effect if the current mode is notes.	
ly:performance-header <i>performance</i>	[Funció]
Return header of performance.	
ly:performance-set-header! <i>performance module</i>	[Funció]
Set the performance header.	
ly:performance-write <i>performance filename name</i>	[Funció]
Write <i>performance</i> to <i>filename</i> storing <i>name</i> as the name of the performance in the file metadata.	
ly:pitch? <i>x</i>	[Funció]
Is <i>x</i> a Pitch object?	
ly:pitch<? <i>p1 p2</i>	[Funció]
Is <i>p1</i> lexicographically smaller than <i>p2</i> ?	
ly:pitch-alteration <i>pp</i>	[Funció]
Extract the alteration from pitch <i>pp</i> .	
ly:pitch-diff <i>pitch root</i>	[Funció]
Return pitch <i>delta</i> such that <i>root</i> transposed by <i>delta</i> equals <i>pitch</i> .	
ly:pitch-negate <i>p</i>	[Funció]
Negate <i>p</i> .	
ly:pitch-notename <i>pp</i>	[Funció]
Extract the note name from pitch <i>pp</i> .	
ly:pitch-octave <i>pp</i>	[Funció]
Extract the octave from pitch <i>pp</i> .	
ly:pitch-quartertones <i>pp</i>	[Funció]
Calculate the number of quarter tones of <i>pp</i> from middle C.	
ly:pitch-semitones <i>pp</i>	[Funció]
Calculate the number of semitones of <i>pp</i> from middle C.	
ly:pitch-steps <i>p</i>	[Funció]
Number of steps counted from middle C of the pitch <i>p</i> .	
ly:pitch-tones <i>pp</i>	[Funció]
Calculate the number of tones of <i>pp</i> from middle C as a rational number.	
ly:pitch-transpose <i>p delta</i>	[Funció]
Transpose <i>p</i> by the amount <i>delta</i> , where <i>delta</i> is relative to middle C.	

ly:pointer-group-interface::add-grob <i>grob sym grob-element</i>	[Funció]
Add <i>grob-element</i> to <i>grob</i> 's <i>sym</i> grob array.	
ly:position-on-line? <i>sg spos</i>	[Funció]
Return whether <i>spos</i> is on a line of the staff associated with the grob <i>sg</i> (even on an extender line).	
ly:prob? <i>x</i>	[Funció]
Is <i>x</i> a Prob object?	
ly:prob-immutable-properties <i>prob</i>	[Funció]
Retrieve an alist of immutable properties.	
ly:prob-mutable-properties <i>prob</i>	[Funció]
Retrieve an alist of mutable properties.	
ly:prob-property <i>prob sym val</i>	[Funció]
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.	
ly:prob-property? <i>obj sym</i>	[Funció]
Is boolean prop <i>sym</i> of <i>sym</i> set?	
ly:prob-set-property! <i>obj sym value</i>	[Funció]
Set property <i>sym</i> of <i>obj</i> to <i>value</i> .	
ly:prob-type? <i>obj type</i>	[Funció]
Is <i>obj</i> the specified prob-type?	
ly:programming-error <i>str rest</i>	[Funció]
A Scheme callable function to issue the internal warning <i>str</i> . The message is formatted with format and <i>rest</i> .	
ly:progress <i>str rest</i>	[Funció]
A Scheme callable function to print progress <i>str</i> . The message is formatted with format and <i>rest</i> .	
ly:property-lookup-stats <i>sym</i>	[Funció]
Return hash table with a property access corresponding to <i>sym</i> . Choices are prob , grob , and context .	
ly:protects	[Funció]
Return hash of protected objects.	
ly:pt <i>num</i>	[Funció]
<i>num</i> printer points.	
ly:pure-call <i>data grob start end rest</i>	[Funció]
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.	
ly:register-stencil-expression <i>symbol</i>	[Funció]
Add <i>symbol</i> as head of a stencil expression.	
ly:relative-group-extent <i>elements common axis</i>	[Funció]
Determine the extent of <i>elements</i> relative to <i>common</i> in the <i>axis</i> direction.	

- ly:reset-all-fonts** [Funció]
Forget all about previously loaded fonts.
- ly:round-filled-box** *xext yext blot* [Funció]
Make a **Stencil** object that prints a black box of dimensions *xext*, *yext* and roundness *blot*.
- ly:round-filled-polygon** *points blot extroversion* [Funció]
Make a **Stencil** object that prints a black polygon with corners at the points defined by *points* (list of coordinate pairs) and roundness *blot*. Optionalextroversion shifts the outline outward, with the default of -1.0 keeping the outer boundary of the outline just inside of the polygon.
- ly:run-translator** *mus output-def* [Funció]
Process *mus* according to *output-def*. An interpretation context is set up, and *mus* 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.
- ly:score? x** [Funció]
Is *x* a **Score** object?
- ly:score-add-output-def!** *score def* [Funció]
Add an output definition *def* to *score*.
- ly:score-embedded-format** *score layout* [Funció]
Run *score* through *layout* (an output definition) scaled to correct output-scale already, returning a list of layout-lines.
- ly:score-error? score** [Funció]
Was there an error in the score?
- ly:score-header score** [Funció]
Return score header.
- ly:score-music score** [Funció]
Return score music.
- ly:score-output-defs score** [Funció]
All output definitions in a score.
- ly:score-set-header!** *score module* [Funció]
Set the score header.
- ly:separation-item::print** [Funció]
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** [Funció]
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* [Funció]
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* [Funció]
Set the `middleCPosition` variable in *context* based on the variables `middleCClefPosition` and `middleCOffset`.
- ly:set-option** *var val* [Funció]
Set a program option.
- ly:set-origin!** *m origin* [Funció]
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* [Funció]
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* [Funció]
Is *x* a Skyline object?
- ly:skyline-empty?** *sky* [Funció]
Return whether *sky* is empty.
- ly:skyline-pair?** *x* [Funció]
Is *x* a Skyline_pair object?
- ly:slur-score-count** [Funció]
count number of slur scores.
- ly:smob-protects** [Funció]
Return LilyPond's internal smob protection list.
- ly:solve-spring-rod-problem** *springs rods length ragged* [Funció]
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* [Funció]
Is *x* a Source_file object?

ly:source-files <i>parser-smob</i>	[Funció]
A list of LilyPond files being processed;a PARSER may optionally be specified.	
ly:spanner? <i>g</i>	[Funció]
Is <i>g</i> a spanner object?	
ly:spanner-bound <i>spanner dir</i>	[Funció]
Get one of the bounds of <i>spanner</i> . <i>dir</i> is -1 for left, and 1 for right.	
ly:spanner-broken-into <i>spanner</i>	[Funció]
Return broken-into list for <i>spanner</i> .	
ly:spanner-set-bound! <i>spanner dir item</i>	[Funció]
Set grob <i>item</i> as bound in direction <i>dir</i> for <i>spanner</i> .	
ly:spawn <i>command rest</i>	[Funció]
Simple interface to <code>g_spawn_sync</code> <i>str</i> . The error is formatted with format and <i>rest</i> .	
ly:spring? <i>x</i>	[Funció]
Is <i>x</i> a Spring object?	
ly:spring-set-inverse-compress-strength! <i>spring strength</i>	[Funció]
Set the inverse compress <i>strength</i> of <i>spring</i> .	
ly:spring-set-inverse-stretch-strength! <i>spring strength</i>	[Funció]
Set the inverse stretch <i>strength</i> of <i>spring</i> .	
ly:staff-symbol-line-thickness <i>grob</i>	[Funció]
Returns the current staff-line thickness in the staff associated with <i>grob</i> , expressed as a multiple of the current staff-space height.	
ly:staff-symbol-staff-radius <i>grob</i>	[Funció]
Returns the radius of the staff associated with <i>grob</i> .	
ly:staff-symbol-staff-space <i>grob</i>	[Funció]
Returns the current staff-space height in the staff associated with <i>grob</i> , expressed as a multiple of the default height of a staff-space in the traditional five-line staff.	
ly:start-environment	[Funció]
Return the environment (a list of strings) that was in effect at program start.	
ly:stderr-redirect <i>file-name mode</i>	[Funció]
Redirect stderr to <i>file-name</i> , opened with <i>mode</i> .	
ly:stencil? <i>x</i>	[Funció]
Is <i>x</i> a Stencil object?	
ly:stencil-add <i>args</i>	[Funció]
Combine stencils. Takes any number of arguments.	
ly:stencil-aligned-to <i>stil axis dir</i>	[Funció]
Align <i>stil</i> using its own extents. <i>dir</i> 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 <i>first axis direction second padding</i>	[Funció]
Construct a stencil by putting <i>second</i> next to <i>first</i> . <i>axis</i> can be 0 (x-axis) or 1 (y-axis). <i>direction</i> can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with <i>padding</i> as extra space. <i>first</i> and <i>second</i> may also be '()' or #f .	









- ly:stencil-empty?** *stil axis* [Funció]
Return whether *stil* is empty. If an optional *axis* is supplied, the emptiness check is restricted to that axis.
- ly:stencil-expr** *stil* [Funció]
Return the expression of *stil*.
- ly:stencil-extent** *stil axis* [Funció]
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* [Funció]
Analyze *s*, and return a list of fonts used in *s*.
- ly:stencil-in-color** *stc r g b* [Funció]
Put *stc* in a different color.
- ly:stencil-rotate** *stil angle x y* [Funció]
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* [Funció]
Return a stencil *stil* rotated *angle* degrees around point (x, y), given in absolute coordinates.
- ly:stencil-scale** *stil x y* [Funció]
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* [Funció]
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* [Funció]
Return a *stil*, but translated by *offset* (a pair of numbers).
- ly:stencil-translate-axis** *stil amount axis* [Funció]
Return a copy of *stil* but translated by *amount* in *axis* direction.
- ly:stream-event?** *obj* [Funció]
Is *obj* a `Stream_event` object?
- ly:string-percent-encode** *str* [Funció]
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* [Funció]
Replace string *a* by string *b* in string *s*.

- ly:system-font-load** *name* [Funció]
 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** [Funció]
 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* [Funció]
 Translates a string in C++ printf format and modifies it to use it for scheme formatting.
- ly:translator?** *x* [Funció]
 Is *x* a Translator object?
- ly:translator-context** *trans* [Funció]
 Return the context of the translator object *trans*.
- ly:translator-description** *me* [Funció]
 Return an alist of properties of translator *me*.
- ly:translator-group?** *x* [Funció]
 Is *x* a Translator_group object?
- ly:translator-name** *trans* [Funció]
 Return the type name of the translator object *trans*. The name is a symbol.
- ly:transpose-key-alist** *l pit* [Funció]
 Make a new key alist of *l* transposed by pitch *pit*.
- ly:truncate-list!** *lst i* [Funció]
 Take at most the first *i* of list *lst*.
- ly:ttf->pfa** *ttf-file-name idx* [Funció]
 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* [Funció]
 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.
- ly:type1->pfa** *type1-file-name* [Funció]
 Convert the contents of a Type 1 font in PFB format to PFA format. If the file is already in PFA format, pass through it.
- ly:undead?** *x* [Funció]
 Is *x* a Undead object?

- ly:unit** [Funció]
Return the unit used for lengths as a string.
- ly:unpure-call** *data grob rest* [Funció]
Convert property *data* (unpure-pure container or procedure) to value in an unpure context defined by *grob* and possibly *rest* arguments.
- ly:unpure-pure-container?** *x* [Funció]
Is *x* a `Unpure_pure_container` object?
- ly:unpure-pure-container-pure-part** *pc* [Funció]
Return the pure part of *pc*.
- ly:unpure-pure-container-unpure-part** *pc* [Funció]
Return the unpure part of *pc*.
- ly:usage** [Funció]
Print usage message.
- ly:verbose-output?** [Funció]
Was verbose output requested, i.e. loglevel at least `DEBUG`?
- ly:version** [Funció]
Return the current lilypond version as a list, e.g., `(1 3 127 uu1)`.
- ly:warning** *str rest* [Funció]
A Scheme callable function to issue the warning *str*. The message is formatted with `format` and *rest*.
- ly:warning-located** *location str rest* [Funció]
A Scheme callable function to issue the warning *str* at the specified location in an input file. The message is formatted with `format` and *rest*.
- ly:wide-char->utf-8** *wc* [Funció]
Encode the Unicode codepoint *wc*, an integer, as UTF-8.

Annex B Cheat sheet

Syntax	Description	Example
<code>1 2 8 16</code>	durations	
<code>c4. c4..</code>	augmentation dots	
<code>c d e f g a b</code>	scale	
<code>f#is b#es</code>	alteration	
<code>\clef treble \clef bass</code>	clefs	
<code>\time 3/4 \time 4/4</code>	time signature	
<code>r4 r8</code>	rest	
<code>d ~ d</code>	tie	
<code>\key es \major</code>	key signature	

`note'`

raise octave

`note,`

lower octave

`c(d e)`

slur

`c\ (c(d) e\)`

phrasing slur

`a8[b]`

beam

`<< \new Staff ... >>`

more staves

`c-> c-.`

articulations

`c2\mf c\s fz`

dynamics

`a\< a a\!`

crescendo



`a\> a a\!`

decrescendo

`< >`

chord

`\partial 8`

pickup / upbeat

`\tuplet 3/2 {f g a}`

triplets

`\grace`

grace notes

`\lyricmode { twinkle }`

entering lyrics

twinkle

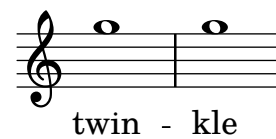
`\new Lyrics`

printing lyrics

twinkle

`twin -- kle`

lyric hyphen

`\chordmode { c:dim f:maj7 }`

chords

`\new ChordNames`

printing chord names

 $C^{\circ} F^{\Delta}$ `<<\{e f\} \\\{c d\}>>`

polyphony



s4 s8 s16

spacer rests

Annex 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.

Annex D Índex d'ordres del LilyPond

Aquest índex fa una relació de totes les ordres i paraules clau del LilyPond, amb enllaços a aquelles seccions del manual que descriuen o s'ocupen del seu ús. Cadascú d'aquests enllaços consta de dues parts. La primera part apunta a la situació exacta del manual en el qual apareix l'ordre o paraula clau; la segona part apunta al començament de la secció corresponent del manual en la qual apareix l'ordre o paraula clau.

!	?
! 6, 122	? 6
"	[
" " 109	[..... 93
,]
' 2] 93
(^
\(..... 133	^ 408
)	_
\) 133	_ 263
,	
, 2 109
—	~
- 119	~ 53
•	A
..... 46	\abs-fontsize 239, 674
/	absolute 773
/ 408	\accent 119
/+ 409	\accepts 586, 587, 588
:	\acciaccatura 112, 773
: 162	\accidentalStyle 28, 773
<	\addChordShape 366, 773
\< 122, 164	addInstrumentDefinition 773
<...> 164	additionalPitchPrefix 412
=	\addlyrics 257, 259, 260
= 9, 130, 783	\addQuote 206, 773
>	\aeolian 22
\> 122, 164	\afterGrace 113, 773
	\aikenHeads 41
	\aikenHeadsMinor 42
	\alias 586
	alignAboveContext 589
	alignBelowContext 278, 589
	\allowPageTurn 541, 773
	allowVoltaHook 773
	\alterBroken 627, 773
	\alternative 147
	annotate-spacing 568

<code>\appendToTag</code>	501, 773
<code>applyContext</code>	774
<code>applyMusic</code>	774
<code>applyOutput</code>	774
<code>\appoggiatura</code>	112, 774
<code>\arpeggio</code>	142
<code>\arpeggioArrowDown</code>	142
<code>\arpeggioArrowUp</code>	142
<code>\arpeggioBracket</code>	142
<code>\arpeggioNormal</code>	142
<code>\arpeggioParenthesis</code>	142
<code>\arpeggioParenthesisDashed</code>	142
<code>\arrow-head</code>	247, 699
<code>\ascendens</code>	439, 446
<code>assertBeamQuant</code>	774
<code>assertBeamSlope</code>	774
<code>\auctum</code>	439, 446
<code>aug</code>	406
<code>\augmentum</code>	446
<code>auto-first-page-number</code>	530
<code>\auto-footnote</code>	721
<code>autoBeaming</code>	84, 571
<code>\autoBeamOff</code>	82, 323
<code>\autoBeamOn</code>	82
<code>\autoBreaksOff</code>	535
<code>\autoBreaksOn</code>	535
<code>\autochange</code>	320, 774
<code>\autoLineBreaksOff</code>	535
<code>\autoLineBreaksOn</code>	535
<code>\autoPageBreaksOff</code>	538
<code>\autoPageBreaksOn</code>	538

B

<code>\backslashed-digit</code>	721
<code>Balloon_engraver</code>	225
<code>\balloonGrobText</code>	225, 774
<code>\balloonLengthOff</code>	225
<code>\balloonLengthOn</code>	225
<code>\balloonText</code>	225, 774
<code>banjo-c-tuning</code>	380
<code>banjo-modal-tuning</code>	380
<code>banjo-open-d-tuning</code>	380
<code>banjo-open-dm-tuning</code>	380
<code>\bar</code>	97, 103, 774
<code>barCheckSynchronize</code>	109
<code>BarNumber</code>	104
<code>\barNumberCheck</code>	109, 774
<code>barNumberVisibility</code>	104
<code>bartype</code>	103
<code>base-shortest-duration</code>	557
<code>baseMoment</code>	84
<code>\beam</code>	699
<code>\beamExceptions</code>	84, 774
<code>beatStructure</code>	84
<code>\bendAfter</code>	136, 774
<code>binding-offset</code>	528
<code>blank-after-score-page-penalty</code>	530
<code>blank-last-page-penalty</code>	530
<code>blank-page-penalty</code>	529
<code>\bold</code>	239, 674
<code>\book</code>	465, 468
<code>\bookOutputName</code>	467, 774
<code>\bookOutputSuffix</code>	467, 774
<code>\bookpart</code>	466, 468, 538

<code>bookTitleMarkup</code>	478
<code>bottom-margin</code>	524
<code>\box</code>	245, 674
<code>\bracket</code>	128, 245, 325, 699
<code>\break</code>	535
<code>breakable</code>	83
<code>\breathe</code>	134, 774
<code>\breve</code>	45, 57

C

<code>\cadenzaOff</code>	73
<code>\cadenzaOn</code>	73
<code>\caesura</code>	438
<code>\caps</code>	675
<code>\cavum</code>	439, 446
<code>\center-align</code>	242, 684
<code>\center-column</code>	244, 684
<code>\change</code>	319
<code>\char</code>	722
<code>check-consistency</code>	527
<code>chordChanges</code>	411
<code>\chordmode</code>	5, 13, 363
<code>chordNameExceptions</code>	414
<code>chordNameLowercaseMinor</code>	412
<code>ChordNames</code>	363
<code>chordNameSeparator</code>	413
<code>chordNoteNamer</code>	413
<code>chordPrefixSpacer</code>	414
<code>\chordRepeats</code>	336, 774
<code>chordRootNamer</code>	412
<code>\chords</code>	410
<code>\circle</code>	245, 700
<code>\clef</code>	17, 774
<code>clip-regions</code>	505
<code>\cm</code>	604
<code>\coda</code>	119
<code>color</code>	222
<code>\column</code>	244, 684
<code>\column-lines</code>	728
<code>\combine</code>	247, 685
<code>common-shortest-duration</code>	557
<code>Completion_heads_engraver</code>	78
<code>Completion_rest_engraver</code>	78
<code>\compound-meter</code>	707
<code>\compoundMeter</code>	77, 775
<code>\compressMMRests</code>	60, 62, 775
<code>\concat</code>	685
<code>\consists</code>	586
<code>\context</code>	573, 581
<code>controlpitch</code>	9
<code>\cr</code>	122
<code>\cresc</code>	124
<code>\crescHairpin</code>	124
<code>\crescTextCresc</code>	124
<code>cross</code>	37
<code>\crossStaff</code>	323, 775
<code>\cueClef</code>	209, 775
<code>cueClefUnset</code>	775
<code>\cueDuring</code>	209, 775
<code>\cueDuringWithClef</code>	209, 775
<code>currentBarNumber</code>	104, 118
<code>\customTabClef</code>	708

D

deadNote.....	775
\decr.....	122
\decresc.....	124
default.....	28, 29
default-staff-staff-spacing.....	542
defaultBarType.....	103
\defaultchild.....	589
defaultNoteHeads.....	775
\defaultTimeSignature.....	65
\defineBarLine.....	101, 775
\deminutum.....	439, 446
\denies.....	586, 587, 588
\descendens.....	439, 446
\dim.....	124, 406
\dimHairpin.....	124
\dimTextDecr.....	124
\dimTextDecresc.....	124
\dimTextDim.....	124
\dir-column.....	685
\discant.....	716
\displayLilyMusic.....	519, 775
displayMusic.....	775
displayScheme.....	775
\divisioMaior.....	438
\divisioMaxima.....	438
\divisioMinima.....	438
dodecaphonic.....	32
dodecaphonic-first.....	33
dodecaphonic-no-repeat.....	33
\dorian.....	22
\dotsDown.....	46
\dotsNeutral.....	46
\dotsUp.....	46
\doubleflat.....	708
\doublesharp.....	708
\downbow.....	119, 329
\downmordent.....	119
\downprall.....	119
\draw-circle.....	247, 700
\draw-dashed-line.....	700
\draw-dotted-line.....	701
\draw-hline.....	701
\draw-line.....	247, 701
\draw-squiggle-line.....	702
\drummode.....	186
DrumStaff.....	186
\dynamic.....	128, 675
\dynamicDown.....	124
DynamicLineSpanner.....	124
\dynamicNeutral.....	124
\dynamicUp.....	124

E

\easyHeadsOff.....	39
\easyHeadsOn.....	39
\ellipse.....	702
endSpanners.....	775
\epsfile.....	247, 703
\espressivo.....	119, 123
\etc.....	632
eventChords.....	775

extra-offset.....	542
\eyeglasses.....	722

F

\f.....	122
\featherDurations.....	96, 775
\fermata.....	119, 708
\fermataMarkup.....	62, 119
\ff.....	122
\fff.....	122
\ffff.....	122
\fffff.....	122
\fill-line.....	244, 686
\fill-with-pattern.....	686
\filled-box.....	247, 703
\finalis.....	438
\finger.....	219, 675, 776
first-page-number.....	530
\first-visible.....	722
\fixed.....	2, 776
\flageolet.....	119
\flat.....	708
\flexa.....	446
followVoice.....	322
font-interface.....	218, 251
font-size.....	215, 218
\fontCaps.....	675
fontSize.....	215
\fontsize.....	239, 676
\footnote.....	483, 722, 776
forget.....	33
four-string-banjo.....	380
\fp.....	122
\fraction.....	722
\freeBass.....	717
\frenchChords.....	414
\fret-diagram.....	353, 713
fret-diagram-interface.....	359
\fret-diagram-terse.....	355, 713
\fret-diagram-verbose.....	357, 714
FretBoards.....	362
\fromproperty.....	723
\funkHeads.....	41
\funkHeadsMinor.....	42

G

\general-align.....	243, 687
\germanChords.....	414
\glissando.....	137
\grace.....	112, 776
GregorianTranscriptionStaff.....	186
Grid_line_span_engraver.....	226
Grid_point_engraver.....	226
gridInterval.....	226
grobdescriptions.....	776
grow-direction.....	96

H

<code>\halfopen</code>	119
<code>\halign</code>	242, 688
<code>\harmonic</code>	330, 339
<code>\harmonicByFret</code>	339, 776
<code>\harmonicByRatio</code>	339, 776
<code>harmonicNote</code>	776
<code>\harmonicsOff</code>	330
<code>\harmonicsOn</code>	330, 776
<code>\harp-pedal</code>	715
<code>\hbracket</code>	245, 703
<code>\hcenter-in</code>	689
<code>\header</code>	468
<code>\hide</code>	611, 776
<code>\hideKeySignature</code>	394
<code>\hideNotes</code>	221
<code>\hideSplitTiedTabNotes</code>	338
<code>\hideStaffSwitch</code>	322
<code>horizontal-shift</code>	528
<code>Horizontal_bracket_engraver</code>	228
<code>\hspace</code>	689
<code>huge</code>	215, 241, 676

I

<code>\improvisationOff</code>	44, 80
<code>\improvisationOn</code>	44, 80
<code>\in</code>	604
<code>\incipit</code>	450, 776
<code>\inclinatum</code>	439, 446
<code>\include</code>	495
<code>indent</code>	203, 528, 561
<code>\inherit-acceptability</code>	587, 776
<code>inner-margin</code>	528
<code>\inStaffSegno</code>	150, 777
<code>instrumentSwitch</code>	777
<code>\inversion</code>	13, 777
<code>\ionian</code>	22
<code>\italianChords</code>	414
<code>\italic</code>	239, 676

J

<code>\justified-lines</code>	250, 728
<code>\justify</code>	244, 691
<code>\justify-field</code>	690
<code>\justify-line</code>	690
<code>\justify-string</code>	691

K

<code>\keepWithTag</code>	498, 777
<code>\key</code>	21, 42, 777
<code>\killCues</code>	213, 777

L

<code>\label</code>	492, 777
<code>\laissezVibrer</code>	55
<code>language</code>	777
<code>languageRestore</code>	777
<code>languageSaveAndChange</code>	777
<code>large</code>	215, 241, 676
<code>\larger</code>	239, 241, 676
<code>last-bottom-spacing</code>	526
<code>\layout</code>	468, 531, 571, 581
<code>layout file</code>	533
<code>layout-set-staff-size</code>	533
<code>\left-align</code>	242, 692
<code>\left-brace</code>	723
<code>\left-column</code>	692
<code>left-margin</code>	527
<code>\lheel</code>	119
<code>\line</code>	692
<code>line-width</code>	526, 561
<code>\linea</code>	439, 446
<code>\lineprall</code>	119
<code>\locrian</code>	22
<code>\longa</code>	45, 57
<code>\longfermata</code>	119
<code>\lookup</code>	723
<code>\lower</code>	242, 693
<code>\ltoe</code>	119
<code>ly:minimal-breaking</code>	539
<code>ly:one-line-auto-height-breaking</code>	540
<code>ly:one-line-breaking</code>	540
<code>ly:optimal-breaking</code>	539
<code>ly:page-turn-breaking</code>	540
<code>\lydian</code>	22
<code>\lyricmode</code>	256, 257
<code>\lyricsto</code>	257, 259

M

<code>m</code>	406
<code>magnification->font-size</code>	215, 533
<code>\magnify</code>	239, 677
<code>magnifyMusic</code>	215, 777
<code>magnifyStaff</code>	777
<code>magstep</code>	215, 533, 604
<code>maj</code>	406
<code>\major</code>	22
<code>major seven symbols</code>	414
<code>majorSevenSymbol</code>	412
<code>make-dynamic-script</code>	128
<code>make-pango-font-tree</code>	253
<code>\makeClusters</code>	169, 777
<code>makeDefaultStringTuning</code>	777
<code>\map-markup-commands</code>	728
<code>\marcato</code>	119
<code>\mark</code>	110, 233, 777
<code>\markalphabet</code>	724
<code>\markLengthOff</code>	70, 234
<code>\markLengthOn</code>	70, 234
<code>\markletter</code>	724
<code>\markup</code>	233, 236, 237
<code>markup-markup-spacing</code>	526
<code>markup-system-spacing</code>	525
<code>\markuplist</code>	236, 250, 251
<code>markupMap</code>	777

max-systems-per-page	529
\maxima	45, 57
measureLength	84, 118
measurePosition	72, 118
\medium	677
\melisma	264
\melismaEnd	264
MensuralStaff	186
\mergeDifferentlyDottedOff	173
\mergeDifferentlyDottedOn	173
\mergeDifferentlyHeadedOff	173
\mergeDifferentlyHeadedOn	173
\mf	122
\midi	468, 571
midiBalance	517
midiChannelMapping	515
midiChorusLevel	517
midiExpression	517
midiPanPosition	517
midiReverbLevel	517
min-systems-per-page	529
minimum-Y-extent	542
minimumFret	336, 374
minimumPageTurnLength	540
minimumRepeatLengthForPageTurn	541
\minor	22
minorChordModifier	414
mixed	325
\mixolydian	22
\mm	604
\modalInversion	16, 778
\modalTranspose	15, 778
modern	30
modern-cautionary	30
modern-voice	30
modern-voice-cautionary	31
\mordent	119
\mp	122
MultiMeasureRestText	62
\musicglyph	111, 708
musicMap	778

N

\name	586
\natural	709
neo-modern	31
neo-modern-cautionary	32
neo-modern-voice	32
neo-modern-voice-cautionary	32
\new	573
\newSpacingSection	558
no-reset	33
\noBeam	93
\noBreak	535
nonstaff-nonstaff-spacing	542
nonstaff-relatedstaff-spacing	542
nonstaff-unrelatedstaff-spacing	542
\noPageBreak	538, 778
\noPageTurn	541, 778
\normal-size-sub	677
\normal-size-super	240, 678
\normal-text	678
normalsize	215, 241, 678
\note	709

\note-by-number	709
Note_heads_engraver	78
\null	242, 724
\number	678
\numericTimeSignature	65

O

\octaveCheck	9, 778
offset	778
\omit	610, 778
\on-the-fly	481, 724
\once	596, 778
\oneVoice	169
\open	119, 329
\oriscus	439, 446
\ottava	24, 778
outer-margin	528
outside-staff-horizontal-padding	555
outside-staff-padding	555
outside-staff-priority	555
\oval	704
\overlay	693
\override	595, 599, 724
\override-lines	729
\overrideProperty	599, 778
\overrideTimeSignatureSettings	66, 778
\overtie	679

P

\p	122
\pad-around	246, 693
\pad-markup	246, 693
\pad-to-box	246, 694
\pad-x	246, 694
page-breaking	529
page-breaking-system-system-spacing	529
page-count	529
\page-link	725
page-number-type	530
\page-ref	492, 725
page-spacing-weight	530
\pageBreak	538, 779
\pageTurn	541, 779
palmMute	779
palmMuteOn	779
\paper	468, 522
paper-height	524
paper-width	526
\parallelMusic	183, 779
\parenthesize	223, 704, 779
\partcombine	178, 284, 779
\partcombineApart	180
\partcombineAutomatic	180
\partcombineChords	180
partcombineDown	779
partcombineForce	779
\partcombineSoloI	180
\partcombineSoloII	180
\partcombineUnisono	180
partcombineUp	779
\partial	72, 147, 149, 779
\path	705

<code>\pattern</code>	725
<code>pedalSustainStyle</code>	325
<code>percent</code>	159
<code>\pes</code>	446
<code>\phrasingSlurDashed</code>	133
<code>\phrasingSlurDashPattern</code>	134, 779
<code>\phrasingSlurDotted</code>	133
<code>\phrasingSlurDown</code>	133
<code>\phrasingSlurHalfDashed</code>	133
<code>\phrasingSlurHalfSolid</code>	133
<code>\phrasingSlurNeutral</code>	133
<code>\phrasingSlurSolid</code>	133
<code>\phrasingSlurUp</code>	133
<code>\phrygian</code>	22
<code>piano</code>	31
<code>piano-cautionary</code>	31
<code>PianoStaff</code>	318, 320
<code>Pitch_squash_engraver</code>	80
<code>\pitchedTrill</code>	146, 780
<code>pointAndClickOff</code>	780
<code>pointAndClickOn</code>	780
<code>pointAndClickTypes</code>	780
<code>\portato</code>	119
<code>\postscript</code>	247, 705
<code>\powerChords</code>	378
<code>\pp</code>	122
<code>\ppp</code>	122
<code>\pppp</code>	122
<code>\ppppp</code>	122
<code>\prall</code>	119
<code>\pralldown</code>	119
<code>\prallmordent</code>	119
<code>\prallprall</code>	119
<code>\prallup</code>	119
<code>\predefinedFretboardsOff</code>	372
<code>\predefinedFretboardsOn</code>	372
<code>print-all-headers</code>	531
<code>print-first-page-number</code>	530
<code>print-page-number</code>	530
<code>\property-recursive</code>	725
<code>propertyOverride</code>	780
<code>propertyRevert</code>	780
<code>propertySet</code>	780
<code>propertyTweak</code>	780
<code>propertyUnset</code>	780
<code>\pt</code>	604
<code>\pushToTag</code>	501, 780
<code>\put-adjacent</code>	694

Q

<code>\quilisma</code>	439, 446
<code>quotedCueEventTypes</code>	208
<code>quotedEventTypes</code>	208
<code>\quoteDuring</code>	206, 209, 780

R

<code>r</code>	57
<code>ragged-bottom</code>	524
<code>ragged-last</code>	527, 561
<code>ragged-last-bottom</code>	524
<code>ragged-right</code>	527, 561
<code>\raise</code>	242, 694
<code>\relative</code>	2, 5, 13, 321, 781
<code>\RemoveAllEmptyStaves</code>	200
<code>\RemoveEmptyStaves</code>	200
<code>\removeWithTag</code>	498, 781
<code>\repeat</code>	147
<code>\repeat percent</code>	159
<code>\repeat tremolo</code>	162
<code>repeatCommands</code>	155
<code>\repeatTie</code>	54, 150, 280
<code>\replace</code>	679
<code>resetRelativeOctave</code>	781
<code>\rest</code>	57, 710
<code>\rest-by-number</code>	710
<code>restrainOpenStrings</code>	336
<code>\retrograde</code>	14, 781
<code>\reverseturn</code>	119
<code>\revert</code>	596
<code>\revertTimeSignatureSettings</code>	67, 781
<code>\rfz</code>	122
<code>rgb-color</code>	223
<code>\rheel</code>	119
<code>R</code>	60
<code>RhythmicStaff</code>	186
<code>\right-align</code>	242, 695
<code>\right-brace</code>	725
<code>\right-column</code>	695
<code>right-margin</code>	527
<code>\rightHandFinger</code>	375, 781
<code>\roman</code>	679
<code>\romanStringNumbers</code>	329
<code>\rotate</code>	695
<code>\rounded-box</code>	245, 706
<code>\rtoe</code>	119

S

<code>s</code>	59
<code>\sacredHarpHeads</code>	41
<code>\sacredHarpHeadsMinor</code>	42
<code>\sans</code>	680
<code>\scale</code>	706
<code>\scaleDurations</code>	53, 75, 781
<code>\score</code>	464, 468, 711
<code>\score-lines</code>	729
<code>score-markup-spacing</code>	525
<code>score-system-spacing</code>	526
<code>scoreTitleMarkup</code>	478
<code>\segno</code>	119
<code>self-alignment-X</code>	542
<code>\semiflat</code>	712
<code>\semiGermanChords</code>	414
<code>\semisharp</code>	712
<code>\sesquiflat</code>	712
<code>\sesquisharp</code>	712
<code>\set</code>	84, 593, 599
<code>set-global-staff-size</code>	533
<code>set-octavation</code>	24

settingsFrom	781
\sf	122
\sff	122
\sfz	122
\shape	624, 781
\sharp	712
shiftDurations	781
\shiftOff	173
\shiftOn	173
\shiftOnn	173
\shiftOnnn	173
short-indent	203, 528
\shortfermata	119
show-available-fonts	253
showFirstLength	505
\showKeySignature	394
showLastLength	505
\showStaffSwitch	322
\signumcongruentiae	119
\simple	680
single	781
\skip	59, 280, 781
skipTypesetting	505
slashChordSeparator	413
\slashed-digit	726
\slashedGrace	112, 781
\slurDashed	131
\slurDashPattern	131, 781
\slurDotted	131
\slurDown	130
\slurHalfDashed	131
\slurHalfSolid	131
\slurNeutral	130
\slurSolid	131
\slurUp	131
small	215, 241, 680
\smallCaps	680
\smaller	239, 241, 680
\snappizzicato	119
\sostenutoOff	325
\sostenutoOn	325
\southernHarmonyHeads	41
\southernHarmonyHeadsMinor	42
\sp	122
spacing	557
spacingTweaks	781
Span_stem_engraver	323
\spp	122
\staccatissimo	119
\staccato	119
staff-affinity	542
staff-staff-spacing	542
Staff_midiInstrument	518
Staff_symbol_engraver	200
staffgroup-staff-spacing	542
start-repeat	155
\startGroup	228
\startStaff	194, 197
\startTrillSpan	145
\stdBass	717
\stdBassIV	718
\stdBassV	719
\stdBassVI	720
Stem	323
stem-spacing-correction	557

\stemDown	224
stemLeftBeamCount	94
\stemNeutral	224
stemRightBeamCount	94
\stemUp	224
\stencil	726
\stopGroup	228
\stopped	119
\stopStaff	194, 197, 200
\stopTrillSpan	145
\storePredefinedDiagram	366, 781
\stringTuning	349, 782
stringTunings	348, 362
\strophae	439, 446
\strut	726
styledNoteHeads	782
\sub	240, 681
suggestAccidentals	433
\super	240, 681
sus	408
\sustainOff	325
\sustainOn	325
system-count	529
system-separator-markup	531
system-system-spacing	526
systems-per-page	529

T

\tabChordRepeats	336, 782
tabChordRepetition	782
\tabFullNotation	335
\table	729
\table-of-contents	494, 730
TabStaff	186, 334
TabVoice	334
\tag	498, 782
\tagGroup	501, 782
\taor	394
teaching	33
teeny	215, 241, 681
\tempo	69
temporary	782
\tenuto	119
text	325, 682
\textLengthOff	62, 231
\textLengthOn	62, 231
\textSpannerDown	232
\textSpannerNeutral	232
\textSpannerUp	232
\thumb	119, 219
\tie	682
\tied-lyric	712
\tieDashed	55
tieDashPattern	782
\tieDotted	55
\tieDown	55
\tieNeutral	55
\tieSolid	55
\tieUp	55
\time	65, 84, 782
times	782
timeSignatureFraction	75
tiny	215, 241, 682
\tocItem	494, 782

top-margin 524
 top-markup-spacing 526
 top-system-spacing 526
 \translate 243, 696
 \translate-scaled 243, 696
 \transparent 726
 \transpose 5, 10, 13, 782
 \transposedCueDuring 212, 782
 \transposition 26, 206, 782
 \treCorde 325
 tremolo 162
 \triangle 247, 707
 \trill 119, 145
 \tuplet 48, 75, 783
 \tupletDown 48
 \tupletNeutral 48
 TupletNumber 49
 tupletNumberFormatFunction 49
 tupletSpan 783
 tupletSpannerDuration 49
 \tupletUp 48
 \turn 119
 \tweak 597, 599, 783
 two-sided 528
 \type 586
 \typewriter 683

U

\unaCorda 325
 \underline 239, 683
 \undertie 683
 undo 783
 unfold 157
 \unfoldRepeats 514, 783
 \unHideNotes 221
 \unset 594
 \upbow 119, 329
 \upmordent 119
 \upprall 119
 \upright 683

V

\varcoda 119
 VaticanaStaff 186
 \vcenter 696
 \verbatim-file 726
 VerticalAxisGroup 542
 \verylongfermata 119
 \virga 439, 446
 \virgula 438
 Voice 169
 voice 28, 29
 \voiceFourStyle 173
 \voiceNeutralStyle 173

\voiceOne 169
 \voiceOne ... \voiceFour 169
 \voiceOneStyle 173
 \voiceThreeStyle 173
 \voiceTwoStyle 173
 \void 519, 783
 \vspace 696

W

\walkerHeads 41
 \walkerHeadsMinor 42
 whichBar 103
 \whiteout 727
 \with 579, 584
 \with-color 222, 727
 \with-dimensions 728
 \with-dimensions-from 728
 \with-link 728
 \with-url 707
 withMusicProperty 783
 \woodwind-diagram 716
 \wordwrap 244, 698
 \wordwrap-field 697
 \wordwrap-internal 730
 \wordwrap-lines 250, 730
 \wordwrap-string 698
 \wordwrap-string-internal 730

X

x11-color 222, 223
 X-offset 542
 xNote 783
 xNotesOn 783

Annex E Índex del LilyPond

A més de totes les ordres i paraules clau del LilyPond, aquest índex és una llista de termes musicals i les paraules que tenen relació amb cada u d'ells, amb enllaços a aquelles seccions del manual que descriuen o s'ocupen d'aquest terme. Cada un dels enllaços consta de dues parts. La primera part apunta a la situació exacta del manual en el qual apareix el terme; la segona part apunta al començament de la secció corresponent del manual en la qual es discuteix aquest terme.

!	>
! 6, 122	\> 122, 164
"	?
" " 109	? 6
,	[
' 2	[..... 93
(]
\(..... 133] 93
)]
\) 133] 93
,	^
, 2	^ 408
—	—
- 119	- 263
•	
..... 46 109
/	~
/ 408	~ 53
/+ 409	
:	1
: 162	15ma 24
<	8
\< 122, 164	8va 24
<...> 164	
=	
= 9, 130, 783	

A

- a due part 178
- `\abs-fontsize` 239, 674
- absoluta 1
- absoluta, especificació, de l'octava 1
- absoluta, introducció, de l'octava 1
- `absolute` 773
- absolute dynamics 122
- `\accent` 119, 120, 732
- accentus 733
- `\accepts` 586, 587, 588
- acciacatura 112, 773
- acciacatura, multi-note 116
- accidental 5
- Accidental, musica ficta 433
- accidentals 433, 437, 448
- accidentals, cadenzas 73
- accidentals, unmetered music 73
- `\accidentalStyle` 28, 773
- accordion 326
- accordion discant symbols 326
- accordion shift symbols 326
- accordion shifts 326
- acordes, alteracions dins de 34
- acords i introduccions relativa de l'octava 5
- acoustic bass 733
- acoustic snare 733
- `\addChordShape` 366, 773
- adding a white background to text 727
- adding custom fret diagrams 365
- `addInstrumentDefinition` 773
- `additionalPitchPrefix` 412
- additions, in chords 407
- `\addlyrics` 257, 259, 260
- `\addQuote` 206, 773
- adjusting staff symbol 605
- `\aeolian` 22
- `\afterGrace` 113, 773
- `afterGraceFraction` 738
- agogo 733
- Aiken, caps de nota amb forma 41
- `\aikenHeads` 41
- `\aikenHeadsMinor` 42
- al niente 125
- `\alias` 586
- align to objects 621
- `alignAboveContext` 589
- `alignBelowContext` 278, 589
- aligning markup text 242
- aligning markups 242
- aligning text 242
- aligning to cadenza 117
- alignment, text, commands 245
- alist 736
- `\allowPageTurn` 541, 773
- `allowVoltaHook` 773
- alteració con parèntesi 6
- alteració de precaució 6
- alteració de quarts de to 7
- alteració de veu de precaució modern, estil de 31
- alteració en nota lligada 6
- alteració recordatòria 6
- alteració, estil de veu de 29
- alteració, estil de veu de precaució modern 31
- alteració, estil de, oblit 33
- alteració, estil modern de 30
- alteració, modern de precaució, veu, estil de 31
- alteracions accidentals 28
- alteracions automàtiques 28
- alteracions de piano 31
- alteracions en acordes 34
- alteracions estil modern 29
- alteracions i notes simultànies 34
- alteracions modernes 30
- alteracions sense restabliment, estil de 33
- alteracions, en diverses veus 30
- alteracions, estil de las 28
- alteracions, estil modern 30
- alteracions, estil modern de precaució 30
- alteracions, estil predeterminat 29
- alteracions, estil teaching (ensenyament) 33
- alteracions, estil, dodecafònic 32
- alteracions, estil, modern-precaució 29
- alteracions, estil, neo-modern de veu 32
- alteracions, estil, neo-modern-voice-cautionary 32
- alteracions, estilo de, neo-modern 31
- alteracions, estilo predeterminat de les 28
- alteracions, estilo, neo-moderno de precaució 32
- alteracions, piano de precaució 31
- alteracions, piano de precaució, estilo de 31
- `\alterBroken` 627, 773
- altered chords 407
- alternate endings 147
- alternate endings, repeats 157
- alternate repeats 157
- `\alternative` 147
- alternative endings and lyrics 279
- alternative endings, with ties 150
- alternative melody, switching to 289
- alto, clau de 17
- altures 1
- altures, noms de, altres llengües 8
- altures, transport de 10
- Amazing Grace bagpipe example 394
- ambitus 35
- anacrusis 72
- anacrusis in a repeat 149
- analysis, musicological 228
- angled hairpins 617
- `annotate-spacing` 568
- anthems 294
- antigues, claus 17
- `\appendToTag` 501, 773
- `applyContext` 774
- `applyMusic` 774
- `applyOutput` 774
- appoggiatura 112, 774
- Arabic key signatures 459
- Arabic music 457
- Arabic music example 461
- Arabic music template 461
- Arabic note names 458
- Arabic semi-flat symbol 458
- Arabic time signatures 460
- Armadura de la tonalitat 21
- arpa sacra, caps de nota amb forma 41
- arpeggio 142
- arpeggio symbols, special 142
- arpeggio, cross-staff parenthesis-style 145

arpeggio, parenthesis-style, cross-staff	145
<code>\arpeggioArrowDown</code>	142
<code>\arpeggioArrowUp</code>	142
<code>\arpeggioBracket</code>	142
<code>\arpeggioNormal</code>	142
<code>\arpeggioParenthesis</code>	142
<code>\arpeggioParenthesisDashed</code>	142
<code>\arrow-head</code>	247, 699
articulate script	518
articulate.ly	518
articulation-event	208
articulations	119, 438
artificial harmonics	330
<code>\ascendens</code>	439, 446
aspes, caps de nota	37
<code>assertBeamQuant</code>	774
<code>assertBeamSlope</code>	774
<code>associatedVoice</code>	257, 259, 289
association list	736
<code>\auctum</code>	439, 446
<code>aug</code>	406
<code>\augmentum</code>	446
<code>auto-first-page-number</code>	530
<code>\auto-footnote</code>	721
<code>autoBeaming</code>	84, 571
autobeaming properties for time signatures	66
<code>\autoBeamOff</code>	82, 323
<code>\autoBeamOn</code>	82
<code>\autoBreaksOff</code>	535
<code>\autoBreaksOn</code>	535
<code>\autochange</code>	320, 774
autochange and relative music	321
<code>\autoLineBreaksOff</code>	535
<code>\autoLineBreaksOn</code>	535
automàtiques, alteracions	28
automatic chord diagrams	372
automatic fret diagrams	372
automatic part combining	178
automatic staff changes	320
automaticBars	616
<code>\autoPageBreaksOff</code>	538
<code>\autoPageBreaksOn</code>	538
available fonts, listing	253

B

Bézier curves, control points	624
backslashed digits	721
<code>\backslashed-digit</code>	721
bagpipe	394
bagpipe example	394
baix, clau de	17
Balance in MIDI	517
balloon	225
balloon help	225
<code>Balloon_engraver</code>	225
<code>\balloonGrobText</code>	225, 774
<code>\balloonLengthOff</code>	225
<code>\balloonLengthOn</code>	225
<code>\balloonText</code>	225, 774
banjo tablature	332
banjo tablatures	380
banjo tunings	380
<code>banjo-c-tuning</code>	380
<code>banjo-modal-tuning</code>	380

<code>banjo-open-d-tuning</code>	380
<code>banjo-open-dm-tuning</code>	380
<code>\bar</code>	97, 103, 774
bar check	109
bar checks with repeats	149
bar lines	97
bar lines, cadenzas	73
bar lines, closing	97
bar lines, default, changing	103
bar lines, defining	101
bar lines, double	97
bar lines, invisible	97
bar lines, manual	97
bar lines, suppressing	616
bar lines, symbols on	233
bar lines, unmetred music	73
bar number	118
bar number alignment	108
bar number check	109
bar number collision	109
bar number, format	106
bar numbers	104
bar numbers, cadenzas	73
bar numbers, regular spacing	104
bar numbers, unmetred music	73
bar numbers, with letters	107
bar numbers, with repeats	107
baríton, clau de	17
<code>barCheckSynchronize</code>	109
<code>BarNumber</code>	104
<code>\barNumberCheck</code>	109, 774
<code>barNumberVisibility</code>	104
barrades, capss de nota	44
barre indications	353
Bartók pizzicato	331
<code>bartype</code>	103
<code>base-shortest-duration</code>	557
<code>baseMoment</code>	84
bass	733
bass note, for chords	408
Bass, figured	417
Bass, thorough	417
Basso continuo	417
<code>\beam</code>	699
beam, endings in a score	91
beam, endings with multiple voices	91
<code>\beamExceptions</code>	84, 774
beaming, time signature default properties	66
beamlets, orienting	90
beams, cadenzas	73
beams, cross-staff	319
beams, customizing rules	82
beams, feathered	96
beams, line breaks	83
beams, manual	82, 93
beams, <code>\partcombine</code> with <code>\autoBeamOff</code>	83
beams, subdividing	89
beams, unmetred music	73
beams, with knee gap	83
beams, with lyrics	84
beams, with melismata	82
beams, with polymetric meters	75
beats per minute	69
beats, grouping	90
<code>beatStructure</code>	84

becaire	5
bemoll	5
bemoll, doble	5
<code>\bendAfter</code>	136, 774
binding gutter	528
<code>binding-offset</code>	528
bisbiglando	327
Bison	738
<code>blank-after-score-page-penalty</code>	530
<code>blank-last-page-penalty</code>	530
<code>blank-page-penalty</code>	529
BNF	738
<code>\bold</code>	239, 674
bongo	733
<code>\book</code>	465, 468
<code>\bookOutputName</code>	467, 774
<code>\bookOutputSuffix</code>	467, 774
<code>\bookpart</code>	466, 468, 538
<code>bookTitleMarkup</code>	478
<code>bottom-margin</code>	524
bounding box	604
bowing indications	329
<code>\box</code>	245, 674
brace, vertical	187
braces, nesting of	191
braces, various sizes	251
<code>\bracket</code>	128, 245, 325, 699
bracket, horizontal	228
bracket, phrasing	228
bracket, vertical	187
bracket, volta	155
brackets	223, 228
brackets, angle	164
brackets, nesting of	191
<code>\break</code>	535
<code>break-align-symbols</code>	621
<code>break-visibility</code>	612
<code>breakable</code>	83
<code>breakbefore</code>	475
breaking lines	535
breaking pages	561
breaks in unmeasured music	75
breath marks	134
<code>\breathe</code>	134, 774
<code>\breve</code>	45, 57
breve rest	57
broken chord	142
broken spanners, modifying	627

C

cabasa	733
cadenza	73, 117
cadenza, accidentals	73
cadenza, aligning to	117
cadenza, bar lines	73
cadenza, bar numbers	73
cadenza, beams	73
cadenza, line breaks	75
cadenza, page breaks	75
<code>\cadenzaOff</code>	73
<code>\cadenzaOn</code>	73
caesura	135, 438
callback	736

capo	357
<code>\caps</code>	675
<code>\cavum</code>	439, 446
<code>\center-align</code>	242, 684
<code>\center-column</code>	244, 684
centered dynamics in piano music	318
centering a column of text	684
centering text on the page	244
<code>\change</code>	319
changing direction of text columns	685
changing fonts	239
changing instrument names	204
changing properties	593
changing staff automatically	320
changing staff manually	319
chants	305
<code>\char</code>	722
character names	298
<code>check-consistency</code>	527
chinese cymbal	733
choir staff	187
chord chords	404
chord diagrams	352, 362
chord diagrams, automatic	372
chord glissandi	346
chord inversions	408
chord mode	404
chord names	404, 409
chord names with fret diagrams	363
chord quality	405
chord shapes for fretted instruments	366
chord steps, altering	408
chord, broken	142
chord, modifying one note in	597
chord, repetition	166, 336
<code>chordChanges</code>	411
<code>\chordmode</code>	5, 13, 363
<code>chordNameExceptions</code>	414
<code>chordNameLowercaseMinor</code>	412
<code>ChordNames</code>	363
<code>chordNameSeparator</code>	413
<code>chordNoteNamer</code>	413
<code>chordPrefixSpacer</code>	414
<code>\chordRepeats</code>	336, 774
<code>chordRootNamer</code>	412
chords	164, 409, 410
chords and ties	54
chords, cross-staff	323
chords, empty	165
chords, fingering	219
chords, jazz	412
chords, power	378
chords, relative pitch	165
chords, splitting across staves	
with <code>\autochange</code>	322
chords, suppressing repeated	411
Chorus level in MIDI	517
Christian Harmony, caps de nota amb forma	41
church rest	62
<code>\circle</code>	245, 700
circling text	700
circulus	733
clau	17
clau antiga	17
clau de alto	17

clau de baix	17	<code>\context</code>	573, 581
clau de bariton	17	context definitions with MIDI	513
clau de contrabaix	17	<code>\context</code> in <code>\layout</code> block	581
clau de Do	17	context properties, changing defaults	581
clau de Fa	17	Contexts, creating and referencing	573
clau de mezzosoprano	17	contexts, defining new	586
clau de Sol	17	contexts, implicit	589
clau de soprano	17	contexts, keeping alive	576
clau de tenor	17	contexts, layout order	588
clau de violí	17	contexts, lifetime	576
clau francesa	17	contrabaix, clau de	17
clau transpositora	17	control points, Bézier curves	624
Claus amb notes guia	17	control points, tweaking	599
Claus de guia	17	control, altures de	9
claves	733	controlling general text alignment	687
clef	5, 17, 774	<code>controlpitch</code>	9
clef, moderntab	351	copyright sign	504
clef, percussion	382	coral, clau de tenor	17
clef, tab	351	cowbell	733
clef, visibility following explicit change	613	<code>\cr</code>	122
clefs	429, 436, 447	crash cymbal	733
clefs, visibility of transposition	616	creating a table	729
<code>clip-regions</code>	505	creating empty text objects	724
closing bar lines	97	creating horizontal spaces in text	689
closure	736	creating text fractions	722
cluster	169	creating vertical spaces in text	696, 726
<code>\cm</code>	604	<code>\cresc</code>	124
coda	111, 119, 733	crescendo	122
coda on bar line	233	crescendo-event	208
collision, bar number	109	<code>\crescHairpin</code>	124
collisions	173	<code>\crescTextCresc</code>	124
collisions, cross-staff voices	319	<code>cross</code>	37
collisions, ignoring	169, 178	cross staff chords	323
<code>color</code>	222	cross staff line	322
color in chords	223	cross staff notes	323
color, rgb	223	cross staff stems	323
colored notes	222	cross-staff	322
colored notes in chords	223	cross-staff beams	319
colored objects	222	cross-staff chords	323
coloring notes	222	cross-staff collisions	319
coloring objects	222, 611	cross-staff line	322
coloring text	727	cross-staff notes	319, 323
coloring voices	173	cross-staff parenthesis-style arpeggio	145
colors	222	cross-staff stems	323
Colors, list of	650	cross-staff tremolo	163
<code>\column</code>	244, 684	<code>\crossStaff</code>	323, 775
<code>\column-lines</code>	728	cue notes	206, 209
columns, text	244	cue notes, formatting	209
<code>\combine</code>	247, 685	cue notes, removing	213
combining parts	178	<code>\cueClef</code>	209, 775
comma intervals	462	<code>cueClefUnset</code>	775
<code>common-shortest-duration</code>	557	<code>\cueDuring</code>	209, 775
<code>Completion_heads_engraver</code>	78	<code>\cueDuringWithClef</code>	209, 775
<code>Completion_rest_engraver</code>	78	cues, musical	300
compound time signatures	77	<code>CueVoice</code>	209
<code>\compound-meter</code>	707	cuica	733
<code>\compoundMeter</code>	77, 775	<code>currentBarNumber</code>	104, 118
compressing music	53	custodes	427
<code>\compressMMRests</code>	60, 62, 775	custom fret diagrams	352
<code>\concat</code>	685	custom fret diagrams, adding	365
concatenating text	685	custom rehearsal mark	111
condensing rests	64	custom string tunings	349
conga	733	customized fret diagram	359
<code>\consists</code>	586	customizing chord names	412
constante hairpins	126	<code>\customTabClef</code>	708

custos	427
cymbal, various	733

D

dòric, mode	22
D.S. al Fine	111
dampened notes on fretted instruments	377
dashed phrasing slur	133
dashed slur	131
dashed ties	55
deadNote	775
decorating text	245
\decr	122
\decresc	124
decrescendo	122
default	28, 29
default bar lines, changing	103
default context properties, changing	581
default note duration	46
default-staff-staff-spacing	542
defaultBarType	103
\defaultchild	589
defaultNoteHeads	775
\defaultTimeSignature	65
\defineBarLine	101, 775
defining bar lines	101
\deminutum	439, 446
\denies	586, 587, 588
\descendens	439, 446
diagram, fret, customized	359
diagrams, chord for fretted instruments	352
diagrams, fret	352
diagrams, fret, transposing	364
diamond-shaped note heads	330
\dim	124, 406
dimensions	604
\dimHairpin	124
diminuendo	122
\dimTextDecr	124
\dimTextDecresc	124
\dimTextDim	124
\dir-column	685
\discant	716
discant symbols, accordion	326
\displayLilyMusic	519, 775
displayMusic	775
displayScheme	775
distance between staves	541
distances, absolute	604
distances, scaled	604
divided lyrics	283
divisio	437
\divisioMaior	438
\divisioMaxima	438
\divisioMinima	438
divisiones	437
Do, clau de	17
doble bemoll	5
doble sostingut	5
dodecafònic, estil de alteracions	32
dodecaphonic	32
dodecaphonic-first	33
dodecaphonic-no-repeat	33
doits	136
\dorian	22
\dotsDown	46
\dotsNeutral	46
\dotsUp	46
dotted notes	46
dotted phrasing slurs	133
dotted slur	131
dotted ties	55
double bar lines	97
double time signatures	75
double-dotted notes	46
\doubleflat	708
\doublesharp	708
down bow indication	329
downbow	119, 329, 733
\downmordent	119, 732
\downprall	119, 732
\draw-circle	247, 700
\draw-dashed-line	700
\draw-dotted-line	701
\draw-hline	701
\draw-line	247, 701
\draw-squiggle-line	702
drawing a line across a page	701
drawing beams within text	699
drawing boxes with rounded corners	703
drawing boxes with rounded corners around text	706
drawing circles within text	700
drawing dashed lines within text	700
drawing dotted lines within text	701
drawing ellipse around text	702
drawing graphic objects	245
drawing lines within text	701
drawing oval around text	704
drawing paths	705
drawing solid boxes within text	703
drawing squiggled lines within text	702
drawing staff symbol	605
drawing triangles within text	707
drum staff	186
\drummode	186
drums	382, 383
drums, various	733
DrumStaff	186
duration, default	46
durations, of notes	45
durations, scaling	52
\dynamic	128, 675
dynamic marks, multiple on one note	123
dynamic marks, new	128
dynamic-event	208
\dynamicDown	124
DynamicLineSpanner	124
\dynamicNeutral	124
dynamics	122
Dynamics in MIDI	509
dynamics, absolute	122
dynamics, centered in keyboard music	318
dynamics, editorial	128
dynamics, parenthesis	128
dynamics, vertical positioning	124
\dynamicUp	124

E

<code>\easyHeadsOff</code>	39
<code>\easyHeadsOn</code>	39
editorial dynamics	128
Effects in MIDI	517
electric snare	733
<code>\ellipse</code>	702
embedded graphics	247
embedding graphic objects	245
empty staves	200
encapsulated postscript output	506
enclosing text in a box with rounded corners	706
enclosing text within a box	674
end repeat	155
<code>endSpanners</code>	775
engravers, including in contexts	586
entering lyrics	256
eoli, mode	22
eoni, mode	22
EPS output	506
<code>\epsfile</code>	247, 703
especials, caps de figura	37
espressivo	119, 123, 732
espressivo articulation	123
estil accidental dodecafònic	33
estil dodecafònic, neo-modern	33
estils de cap de nota	37
<code>\etc</code>	632
<code>eventChords</code>	775
exceptions, chord names	414
expanding music	53
explicitClefVisibility	613
explicitKeySignatureVisibility	613
Expression in MIDI	517
expressions, markup	237
extended chords	407
extender	267
<code>extra-offset</code>	542
<code>\eyeglasses</code>	722

F

<code>\f</code>	122
fàcil, notació, caps de nota de	39
Fa, clau de	17
falls	136
<code>\featherDurations</code>	96, 775
feathered beams	96
fermata	111, 119, 708, 732
fermata on bar line	233
fermata on multi-measure rest	62
<code>\fermataMarkup</code>	62, 119
Ferneyhough hairpins	126
Feta font	651
<code>\ff</code>	122
<code>\fff</code>	122
<code>\ffff</code>	122
<code>\fffff</code>	122
figura, cap de la	37
Figured bass	417
figured bass alignment	422
figured bass extender lines	420
<code>\fill-line</code>	244, 686
<code>\fill-with-pattern</code>	686

<code>\filled-box</code>	247, 703
finalis	437, 438
finding available fonts	253
<code>\finger</code>	219, 675, 776
finger change	219
fingerings	219
fingerings chords	219
fingerings instructions for chords	219
fingerings vs. string numbers	332
fingerings and multi-measure rests	64
fingerings, adding to fret diagrams	373
fingerings, right hand for fretted instruments	375
<code>first-page-number</code>	530
<code>\first-visible</code>	722
<code>\fixed</code>	2, 776
flageolet	119, 733
flags	432
flared hairpins	126
<code>\flat</code>	708
Flex	737
<code>\flexa</code>	446
floor tom tom	733
follow voice	322
<code>followVoice</code>	322
font	737
font families	241
font families, setting	253
font size	239
font size (notation)	215
font size (notation) scaling	215
font size (notation), standard	218
font size, setting	533
font switching	239
Font, Feta	651
<code>font-interface</code>	218, 251
<code>font-size</code>	215, 218
<code>\fontCaps</code>	675
fonts, changing for entire document	253
fonts, explained	251
fonts, finding available	253
fonts, non-text in markup	251
<code>\fontsize</code>	239, 676
<code>fontSize</code>	215
foot marks	119
footers	470
<code>\footnote</code>	483, 722, 776
footnotes	483
footnotes in music expressions	483
footnotes in stand-alone text	488
footnotes, event-based	484
footnotes, time-based	485
<code>forget</code>	33
format, rehearsal mark	111
formatting in lyrics	256
formatting text spanners	231
formatting, cue notes	209
formes, notes con	41
four bar music	537
<code>four-string-banjo</code>	380
<code>\fp</code>	122
<code>\fraction</code>	722
fragments	209
Fragments, music	505
fragments, quoting	206
framing text	245

francesa, clau.....	17
<code>\freeBass</code>	717
<code>\frenchChords</code>	414
Frenched score.....	200
Frenched staff.....	200
Frenched staves.....	197
fret.....	336
fret diagram, customized.....	359
fret diagrams.....	352, 362
fret diagrams with chord names.....	363
fret diagrams, adding custom.....	365
fret diagrams, adding fingerings.....	373
fret diagrams, automatic.....	372
fret diagrams, custom.....	352
fret diagrams, mandolin.....	362
fret diagrams, transposing.....	364
fret diagrams, ukulele.....	362
<code>\fret-diagram</code>	353, 713
fret-diagram markup.....	353
<code>fret-diagram-interface</code>	359
<code>\fret-diagram-terse</code>	355, 713
fret-diagram-terse markup.....	355
<code>\fret-diagram-verbose</code>	357, 714
fret-diagram-verbose markup.....	357
FretBoards	362
fretted instruments, chord shapes.....	366
fretted instruments, dampened notes.....	377
fretted instruments, harmonics.....	377
fretted instruments, indicating position and barring.....	376
fretted instruments, predefined string tunings....	348
fretted instruments, right hand fingerings.....	375
frigi, mode.....	22
<code>\fromproperty</code>	723
full-measure rests.....	60
Funk, caps de nota amb forma.....	41
<code>\funkHeads</code>	41
<code>\funkHeadsMinor</code>	42

G

<code>\general-align</code>	243, 687
<code>\germanChords</code>	414
ghost notes.....	223
glissandi and repeats.....	154
glissando.....	137
global variable.....	738
glyph.....	737
glyphs, music.....	111
<code>\grace</code>	112, 776
grace notes.....	112, 394
grace notes and lyrics.....	288
grace notes, changing layout settings.....	114
grace notes, following.....	113
grace notes, tweaking.....	114
grace-note synchronization.....	116
grammar for LilyPond.....	738
grand staff.....	187
graphic notation.....	247
graphic objects, drawing.....	245
graphic objects, embedding.....	245
graphical object interfaces.....	737
graphical objects.....	737
graphics, embedding.....	245, 247
Gregorian square neumes ligatures.....	439

Gregorian transcription staff.....	186
GregorianTranscriptionStaff	186
grid lines.....	226
<code>Grid_line_span_engraver</code>	226
<code>Grid_point_engraver</code>	226
<code>gridInterval</code>	226
grob.....	591, 737
grob properties.....	595
grob-interface.....	737
grobdescriptions	776
grobs, overwriting.....	611
grobs, visibility of.....	610
grouping beats.....	90
grow-direction	96
guiro.....	733
guitar chord charts.....	80
guitar strumming rhythms, showing.....	80
guitar tablature.....	332
guitarra, caps de nota.....	37
gutter.....	528

H

hairpin.....	122
hairpins at bar lines.....	125
hairpins, angled.....	617
hairpins, constante.....	126
hairpins, Ferneyhough.....	126
hairpins, flared.....	126
Hal Leonard.....	39
half-open high hat.....	733
<code>\halfopen</code>	119, 733
<code>\halign</code>	242, 688
hammer on.....	347
handclap.....	733
harmonics, caps de nota.....	37
<code>\harmonic</code>	330, 339
harmonic indications in tablature notation.....	339
Harmonica Sacra, caps de nota amb forma.....	41
<code>\harmonicByFret</code>	339, 776
<code>\harmonicByRatio</code>	339, 776
harmonicNote	776
harmonics on fretted instruments.....	377
harmonics, artificial.....	330
harmonics, natural.....	330
<code>\harmonicsOff</code>	330
<code>\harmonicsOn</code>	330, 776
harp pedal diagrams.....	327
harp pedals.....	327
<code>\harp-pedal</code>	715
harps.....	327
<code>\hbracket</code>	245, 703
<code>\hcenter-in</code>	689
<code>\header</code>	468
headers.....	470
help, balloon.....	225
hidden notes.....	221
<code>\hide</code>	611, 776
<code>\hideKeySignature</code>	394
<code>\hideNotes</code>	221
<code>\hideSplitTiedTabNotes</code>	338
<code>\hideStaffSwitch</code>	322
hiding ancient staves.....	200
hiding rhythmic staves.....	200
hiding staves.....	200

hiding vaticana staves	200
high bongo	733
high conga	733
high hat	733
high timbale	733
high tom tom	733
horizontal bracket	228
horizontal spacing	556
horizontal spacing, overriding	629
horizontal text alignment	242
<code>horizontal-shift</code>	528
<code>Horizontal_bracket_engraver</code>	228
horizontally centering text	684
<code>\hspace</code>	689
hufnagel	425, 426
huge	215, 241, 676
hymns	305
hyphens	267

I

ictus	733
images, embedding	247
immutable objects	737
immutable properties	737
implicit contexts	589
importing stencils into text	726
improvisació	44
<code>\improvisationOff</code>	44, 80
<code>\improvisationOn</code>	44, 80
<code>\in</code>	604
<code>\incipit</code>	450, 776
incipits, adding	450
<code>\inclinatum</code>	439, 446
<code>\include</code>	495
include-settings	502
including files	495
indent	203, 528, 561
indicating No Chord in ChordNames	410
indicating position and barring for fretted instruments	376
<code>\inherit-acceptability</code>	587, 776
inlining an Encapsulated PostScript image	703
inner-margin	528
inserting music into text	711
inserting PostScript directly into text	705
inserting URL links into text	707
<code>\inStaffSegno</code>	150, 777
instrument names	202, 518
instrument names, adding to other contexts	204
instrument names, centering	203
instrument names, changing	204
instrument names, complex	202
instrument names, short	202
instruments transpositors	11
<code>instrumentSwitch</code>	777
interface	737
interface, layout	591
interleaved music	183
Internals Reference	571
inversió	13
inversió modal	16
<code>\inversion</code>	13, 777
invisible notes	221
invisible rest	59

invisible stem	224
<code>\ionian</code>	22
<code>\italianChords</code>	414
<code>\italic</code>	239, 676

J

jazz chords	412
justified text	244
<code>\justified-lines</code>	250, 728
<code>\justify</code>	244, 691
<code>\justify-field</code>	690
<code>\justify-line</code>	690
<code>\justify-string</code>	691
justifying lines of text	728
justifying text	691

K

keep tagged music	498
<code>\keepWithTag</code>	498, 777
<code>\key</code>	21, 42, 777
key signature	5, 433, 437
key signature, visibility following explicit change	613
keyboard instrument staves	318
keyboard music, centering dynamics	318
keyed instrument staves	318
KievanStaff	446
KievanVoice	446
<code>\killCues</code>	213, 777
kirchenpausen	62
knee gap, with beams	83

L

<code>\label</code>	492, 777
laissez vibrer	55
<code>\laissezVibrer</code>	55
language	777
languageRestore	777
languageSaveAndChange	777
large	215, 241, 676
<code>\larger</code>	239, 241, 676
last-bottom-spacing	526
layers	611
<code>\layout</code>	468, 531, 571, 581
layout file	533
layout interface	591
layout objects	737
<code>layout-set-staff-size</code>	533
ledger lines	194
ledger lines, internal	194
ledger lines, modifying	194
left aligning text	692
<code>\left-align</code>	242, 692
<code>\left-brace</code>	723
<code>\left-column</code>	692
left-margin	527
length of notes	45
lexer	737
<code>\heel</code>	119, 733
lidi, mode	22
Ligatures	427, 449

ligatures in text	685
LilyPond grammar	738
<code>\line</code>	692
line breaks	97
line breaks, beams	83
line breaks, cadenzas	75
line breaks, unmetered music	75
line, cross-staff	322
line, staff-change	322
line, staff-change follower	322
<code>line-width</code>	526, 561
<code>\linea</code>	439, 446
<code>\lineprall</code>	119, 732
lines, grid	226
lines, vertical between staves	226
List of colors	650
listing available fonts	253
llengua, altures en altres	8
llengua, noms de nota en altres	8
lligadura, alteracions i	6
locri, mode	22
<code>\locrian</code>	22
<code>\longa</code>	45, 57
longa rest	57
<code>\longfermata</code>	119, 732
<code>\lookup</code>	723
low bongo	733
low conga	733
low timbale	733
low tom tom	733
<code>\lower</code>	242, 693
lowering text	693
<code>\ltoe</code>	119, 733
lute tablatures	380
lute tunings	381
<code>ly:minimal-breaking</code>	539
<code>ly:one-line-auto-height-breaking</code>	540
<code>ly:one-line-breaking</code>	540
<code>ly:optimal-breaking</code>	539
<code>ly:page-turn-breaking</code>	540
<code>\lydian</code>	22
<code>\lyricmode</code>	256, 257
lyrics and melodies	259
lyrics and tied notes	280
lyrics assigned to one voice	169
lyrics on grace notes	288
lyrics punctuation	256
lyrics, aligning to a melody	257
lyrics, aligning with sporadic melody	577
lyrics, divided	283
lyrics, entering	256
lyrics, formatting	256
Lyrics, increasing space between	273
lyrics, keeping inside margin	231
lyrics, positioning	269
lyrics, repeating	275
lyrics, repeats with alternative endings	279
lyrics, shared among voices	284
lyrics, skip	59
lyrics, skipping notes	280
lyrics, using variables	267
lyrics, with beams	84
<code>\lyricsto</code>	257, 259

M

m	406
música para principiants	39
<code>magnification->font-size</code>	215, 533
<code>\magnify</code>	239, 677
magnifying text	677
<code>magnifyMusic</code>	215, 777
<code>magnifyStaff</code>	777
<code>magstep</code>	215, 533, 604
maj	406
<code>\major</code>	22
major seven symbols	414
major, mode	22
<code>majorSevenSymbol</code>	412
makam	462
makamlar	462
make-dynamic-script	128
<code>make-pango-font-tree</code>	253
<code>\makeClusters</code>	169, 777
<code>makeDefaultStringTuning</code>	777
manual bar lines	97
manual beams	82, 93
manual beams, direction shorthand for	93
manual beams, grace notes	93
manual line breaks	535
manual measure lines	97
manual rehearsal mark	111
manual repeat mark	155
manual staff changes	319
<code>\map-markup-commands</code>	728
maqam	457
maqams	457
maracas	733
<code>\marcato</code>	119, 120, 732
margin, text running over	231
<code>\mark</code>	110, 233, 777
mark, phrasing	133
mark, rehearsal	110
mark, rehearsal, format	111
mark, rehearsal, manual	111
mark, rehearsal, style	111
<code>\markalphabet</code>	724
<code>\markLengthOff</code>	70, 234
<code>\markLengthOn</code>	70, 234
<code>\markletter</code>	724
marks, text	233
<code>\markup</code>	233, 236, 237
markup expressions	237
markup mode, quoted text	238
markup mode, special characters	238
markup on multi-measure rest	62
markup syntax	237
markup text	237
markup text alignment commands	245
markup text padding	246
markup text, aligning	242
markup text, decorating	245
markup text, framing	245
markup text, justified	244
markup text, multi-page	250
markup text, wordwrapped	244
markup, centering on the page	244
markup, conditional	481
markup, multi-line	244

markup, multi-page	250	MIDI equalization	509
markup, music notation inside	248	MIDI metadata	482
markup, score inside	249	MIDI Tracks	515
markup-markup-spacing	526	MIDI using repeats	514
markup-system-spacing	525	MIDI volume	509
\markuplist	236, 250, 251	MIDI, instruments	518
markupMap	777	MIDI, Supported notation	507
markups, aligning	242	MIDI, transposició	26
max-systems-per-page	529	MIDI, Unsupported notation	508
\maxima	45, 57	midiBalance	517
maxima rest	57	midiChannelMapping	515
measure check	109	midiChorusLevel	517
measure groupings	90	midiExpression	517
measure lines	97	midiPanPosition	517
measure lines, invisible	97	midiReverbLevel	517
measure lines, manual	97	min-systems-per-page	529
measure number	118	minimum-Y-extent	542
measure number and repeats	154	minimumFret	336, 374
measure number check	109	minimumPageTurnLength	540
measure number, format	106	minimumRepeatLengthForPageTurn	541
measure numbers	104	\minor	22
measure repeats	159	minorChordModifier	414
measure sub-grouping	90	mirroring markup	706
measure, partial	72	mixed	325
measure, pickup	72	mixolidi, mode	22
measureLength	84, 118	\mixolydian	22
measurePosition	72, 118	\mm	604
Medicaea, Editio	425, 426	modal, inversió	16
\medium	677	modal, transposició	15
medium intervals	457	\modalInversion	16, 778
melisma	264, 267	modals, transformacions	14
\melismaEnd	264	\modalTranspose	15, 778
melismata	264	mode	738
melismata, with beams	82	modern	30
melody rhythms, showing	79	modern de precaució, alteracions d'estil	30
menor, mode	22	modern, alteracions d'estil	30
mensural	425, 426	modern, estil de alteracions	29
Mensural ligatures	434	modern-cautionary	30
mensural music, transcription of	190	modern-precaució, estil d'alteracions	29
MensuralStaff	186, 428	modern-voice	30
MensuralVoice	428	modern-voice-cautionary	31
mensuration sign	430	moderntab clef	351
mensurstriche layout	190	modes	22
\mergeDifferentlyDottedOff	173	modes eclesiàstics	22
\mergeDifferentlyDottedOn	173	modifiers, in chords	405
\mergeDifferentlyHeadedOff	173	mordent	119, 732
\mergeDifferentlyHeadedOn	173	mordent, down	119
merging notes	173	mordent, up	119
merging text	685, 693	movements, multiple	465
meter	65	\mp	122
meter style	65	multi-line markup	244
meter, polymetric	75	multi-line text	244
metronome mark	69	multi-measure rest with markup	62
metronome marking with text	69	multi-measure rest, attaching fermata	62
mezzosoprano, clau de	17	multi-measure rest, attaching text	62
\mf	122	multi-measure rest, contracting	61
micro-tones, tab	351	multi-measure rest, expanding	61
microtons	9	multi-measure rest, script	62
mid tom tom	733	multi-measure rests	60
\midi	468, 571	multi-measure rests and fingerings	64
MIDI	26, 507	multi-measure rests, positioning	63
MIDI block	508	multi-note acciaccatura	116
MIDI Channels	515	multi-page markup	250
MIDI context definitions	513	MultiMeasureRestText	62
MIDI dynamics	509	multiple dynamic marks on one note	123

multiple phrasing slurs	133
multiple slurs	130
multiple voices	173
Music fragments	505
music glyphs	111
music inside markup	248
music, unmetered	118
Musica ficta	433
musical cues	300
<code>\musicglyph</code>	111, 708
<code>musicMap</code>	778
musicological analysis	228
<code>musicQuotes</code>	738
mutable objects	738
mutable properties	738
mute bongo	733
mute conga	733
mute timbale	733

N

N.C. symbol	410
<code>\name</code>	586
name of singer	287
names, character	298
<code>\natural</code>	709
natural harmonics	330
natural, nota	5
<code>neo-modern</code>	31
neo-modern de precaució, estilo de alteracions	32
neo-modern, estilo de alteracions	31
<code>neo-modern-cautionary</code>	32
<code>neo-modern-voice</code>	32
neo-modern-voice, estil d'alteració	32
<code>neo-modern-voice-cautionary</code>	32
neo-modern-voice-cautionary, estil de alteracions ..	32
neomensural	426
nested repeat	154
nested staff brackets	191
nesting of staves	191
<code>\new</code>	573
new contexts	573
new dynamic marks	128
new spacing section	558
new staff	186
<code>\newSpacingSection</code>	558
niente, al	125
no chord symbol	410
<code>no-reset</code>	33
<code>\noBeam</code>	93
<code>\noBreak</code>	535
noms de les notes	1
non-ASCII characters	502
non-empty texts	230
non-musical symbols	247
non-text fonts in markup	251
<code>nonstaff-nonstaff-spacing</code>	542
<code>nonstaff-relatedstaff-spacing</code>	542
<code>nonstaff-unrelatedstaff-spacing</code>	542
<code>\noPageBreak</code>	538, 778
<code>\noPageTurn</code>	541, 778
normal repeat	147
<code>\normal-size-sub</code>	677
<code>\normal-size-super</code>	240, 678
<code>\normal-text</code>	678
<code>normalsize</code>	215, 241, 678
nota, cap de, en aspes	37
nota, cap de, en parlató	37
nota, cap de, en rombe	37
nota, cap de, estils	37
nota, cap de, guitarra	37
nota, cap de, harmònics	37
nota, cap especial de	37
notation font size	215
notation inside markup	248
notation, explaining	225
notation, graphic	247
<code>\note</code>	709
note cluster	169
note collisions	173
note duration, default	46
note durations	45
note grouping bracket	228
note head styles	672
note heads	215
note heads, ancient	431, 447
note heads, diamond-shaped	330
note lengths	45
<code>\note-by-number</code>	709
note-event	208
<code>Note_heads_engraver</code>	78
notes within text by log and dot-count	709
notes within text by string	709
notes, caps de, Aiken	41
notes, caps de, arpa sacra	41
notes, caps de, barrades	44
notes, caps de, Christian Harmony	41
notes, caps de, con formes	41
notes, caps de, estudio	39
notes, caps de, Funk	41
notes, caps de, Harmonica Sacra	41
notes, caps de, improvisació	44
notes, caps de, notació simplificada	39
notes, caps de, para practicar	39
notes, caps de, Southern Harmony	41
notes, caps de, Walker	41
notes, colored	222
notes, colored in chords	223
notes, cross-staff	319, 323
notes, dotted	46
notes, double-dotted	46
notes, ghost	223
notes, hidden	221
notes, invisible	221
notes, noms de, en altres llengües	8
notes, noms holandesos de	5
notes, noms predeterminats	5
notes, parenthesized	223
notes, smaller	209
notes, spacing horizontally	558
notes, splitting	78
notes, transparent	221
notes, transport de	10
<code>\null</code>	242, 724
<code>NullVoice</code>	284
<code>\number</code>	678
numbers, bar	104
numbers, measure	104
<code>\numericTimeSignature</code>	65

O

objects, colored	222
objects, coloring	611
objects, overwriting	611
objects, rotating	617
objects, visibility of	610
oblit, estil de alteracions	33
octava	24
octava, comprovació de	9
octava, correcció de	9
octava, especificació absoluta	1
octava, especificació relativa	2
octava, introducció absoluta	1
octava, introducció relativa	2
octava, marca de canvi de	2
octava, transposició	17
octava, transposició, opcional	17
\octaveCheck	9, 778
offset	778
\omit	610, 778
on-the-fly	481, 724
\once	595, 596, 778
\oneVoice	169
open	119, 329, 733
open bongo	733
open conga	733
open high hat	733
open string indication	329
open timbale	733
operació d'inversió nmodal	16
operació de inversió	13
operació de retrogradació	14
operació de transposició	15
operacions modals	14
oratorio	294
orchestral strings	328
organ pedal marks	119
\oriscus	439, 446
ornamentation	119
ornaments	112, 119
ossia	197, 201
ottava	24, 778
Ottoman music	462
outer-margin	528
output definitions	571
output-count	738
output-def	738
output-suffix	738
outside-staff-horizontal-padding	555
outside-staff-padding	555
outside-staff-priority	555
\oval	704
\overlay	693
\override	595, 599, 724
\override-lines	729
\overrideProperty	599, 778
overrides, reverting	596
\overrideTimeSignatureSettings	66, 778
overriding for only one moment	596
overriding properties within text markup	724
\overtie	679
overtie-ing text	679
overwriting objects	611

P

\p	122
\pad-around	246, 693
\pad-markup	246, 693
\pad-to-box	246, 694
\pad-x	246, 694
padding	592
padding around text	246
padding text	693
padding text horizontally	694
page breaking, manual	538
page breaks	561
page breaks, cadenzas	75
page breaks, unmetered music	75
page layout	561
page numbers in roman numerals	530
page numbers, auto-numbering	530
page numbers, specify the first	530
page numbers, suppress	530
page size	522
page, orientation	523
page-breaking	529
page-breaking-system-system-spacing	529
page-count	529
\page-link	725
page-number-type	530
\page-ref	492, 725
page-spacing-weight	530
\pageBreak	538, 779
\pageTurn	541, 779
palmMute	779
palmMuteOn	779
Pan position in MIDI	517
Pango	251
\paper	468, 522
paper size	522
paper size, landscape	523
paper size, orientation	523
paper-height	524
paper-width	526
parèntesi, alteració amb	6
parallel music	183
\parallelMusic	183, 779
parentheses	223
\parenthesize	223, 704, 779
parlato	303
parlato, caps de nota	37
parser	738
parser variable	738
part combiner	178
part songs	294
\partcombine	178, 284, 779
\partcombine and lyrics	181, 284
\partcombineApart	180
\partcombineAutomatic	180
\partcombineChords	180
partcombineDown	779
partcombineForce	779
partCombineListener	738
\partcombineSoloI	180
\partcombineSoloII	180
\partcombineUnisono	180
partcombineUp	779
\partial	72, 147, 149, 779

partial measure	72	pizzicato, Bartók	331
<code>\path</code>	705	pizzicato, snap	331
paths, drawing	705	placeholder events	165
<code>\pattern</code>	725	placement of lyrics	269
pause mark	134	placing horizontal brackets around text	703
PDF metadata	482	placing parentheses around text	704
pedal diagrams, harp	327	placing vertical brackets around text	699
pedal high hat	733	<code>pointAndClickOff</code>	780
pedal indication styles	325	<code>pointAndClickOn</code>	780
pedal indication, bracket	325	<code>pointAndClickTypes</code>	780
pedal indication, mixed	325	polymetric meters, with beams	75
pedal indication, text	325	polymetric scores	580
pedal marks, organ	119	polymetric signatures	75
pedal sustain style	325	polyphonic music	173
pedal, sostenuto	325	polyphony, shared lyrics	284
pedal, sustain	325	polyphony, single-staff	169
pedals, harp	327	<code>\portato</code>	119, 120, 732
pedals, piano	325	positioning multi-measure rests	63
<code>pedalSustainStyle</code>	325	postscript	247, 705
<code>percent</code>	159	power chords	378
percent repeats	159	<code>\powerChords</code>	378
percussion	382, 383	<code>\pp</code>	122
percussion clef	382	<code>\ppp</code>	122
percussion staff	186	<code>\pppp</code>	122
<code>\pes</code>	446	<code>\ppppp</code>	122
Petrucci	425, 426	prall	119, 732
phrasing bracket	228	prall, down	119
phrasing marks	133	prall, up	119
phrasing slur	130	<code>\pralldown</code>	119, 732
phrasing slur, dashed	133	prallmordent	119, 732
phrasing slur, defining dash patterns	134	prallprall	119, 732
phrasing slur, dotted	133	<code>\prallup</code>	119, 732
phrasing slur, half solid and half dashed	133	precaució, alteració de	6
phrasing slur, multiple	133	precaució, alteracions de piano de	31
phrasing slur, simultaneous	133	precaució, estilo de alteracions, piano	31
phrasing slurs	133	predefined string tunings for	
phrasing, in lyrics	264	fretted instruments	348
<code>\phrasingSlurDashed</code>	133	<code>\predefinedFretboardsOff</code>	372
<code>\phrasingSlurDashPattern</code>	134, 779	<code>\predefinedFretboardsOn</code>	372
<code>\phrasingSlurDotted</code>	133	predeterminat, estil de las alteracions	28
<code>\phrasingSlurDown</code>	133	predeterminats, noms de nota	5
<code>\phrasingSlurHalfDashed</code>	133	prima volta	147
<code>\phrasingSlurHalfSolid</code>	133	principiants, música per a	39
<code>\phrasingSlurNeutral</code>	133	<code>print-all-headers</code>	531
<code>\phrasingSlurSolid</code>	133	<code>print-first-page-number</code>	530
<code>\phrasingSlurUp</code>	133	<code>print-page-number</code>	530
<code>\phrygian</code>	22	printing chord names	409
piano	31	printing order	611
piano de precaució, alteracions	31	printing reserved characters	238
piano de precaució, estilo de alteracions	31	printing special characters	238
piano music, centering dynamics	318	prob	738
piano pedals	325	properties	593
piano staff	187	properties, grob	595
piano staves	318	property object	738
piano, alteracions de	31	<code>\property-recursive</code>	725
piano, estil de alteracions de	31	<code>propertyOverride</code>	780
<code>piano-cautionary</code>	31	<code>propertyRevert</code>	780
<code>PianoStaff</code>	318, 320	<code>propertySet</code>	780
pickup in a repeat	149	<code>propertyTweak</code>	780
pickup measure	72	<code>propertyUnset</code>	780
<code>Pitch_squash_engraver</code>	80	psalms	305
pitched trill with accidental	146	<code>\pt</code>	604
pitched trills	146	pull off	347
<code>\pitchedTrill</code>	146, 780	punctuation in lyrics	256
pitchnames	738	pure containers, Scheme	629

`\pushToTag` 501, 780
`\put-adjacent` 694
 putting space around text 693

Q

q, chord repetition 166, 336
 quarter-tones, tab 351
 quarts de to 6
 quarts de to, alteracions 7
`\quilisma` 439, 446
 quote, voices 206
 quoted text 230
 quoted text in markup mode 238
`quotedCueEventTypes` 208
`quotedEventTypes` 208
`\quoteDuring` 206, 209, 780
 quotes in lyrics 256
 quotes, in lyrics 263

R

r 57
`ragged-bottom` 524
`ragged-last` 527, 561
`ragged-last-bottom` 524
`ragged-right` 527, 561
 railroad tracks 135
`\raise` 242, 694
 raising text 694
 rang de altures 35
 Ratisbona, Editio 426
 recordatòria, alteració 6
 referencing contexts 573
 referencing page labels in text 728
 referencing page numbers in text 725
 regular line breaks 537
 rehearsal mark format 111
 rehearsal mark style 111
 rehearsal mark, manual 111
 rehearsal marks 110
 relatiu 2
 relatiu, mode, transposició i 5
 relativa, especificació de l'octava 2
 relativa, introducció de l'octava, i acords 5
`\relative` 2, 5, 13, 321, 781
 relative music and autochange 321
 relative pitch, chords 165
 relatives, especificació d'octaves 2
 religious music 305
 removals, in chords 408
 remove tagged music 498
`\RemoveAllEmptyStaves` 200, 784
`\RemoveEmptyStaves` 200, 784
`\removeWithTag` 498, 781
 removing cue notes 213
 renaissance music 190
`\repeat` 147
 repeat and measure number 154
 repeat bars 97
 repeat number, changing 155
`\repeat percent` 159
 repeat timing information 154
`\repeat tremolo` 162

repeat volta, changing 155
 repeat with alternate endings 147
 repeat with anacrusis 149
 repeat with pickup 149
 repeat with upbeat 149
 repeat, ambiguous 154
 repeat, end 155
 repeat, manual 155
 repeat, measure 159
 repeat, nested 154
 repeat, normal 147
 repeat, percent 159
 repeat, short 159
 repeat, start 155
 repeat, tremolo 162
`repeatCommands` 155
 repeating lyrics with alternative endings 279
 repeating ties 54
 repeats 99
 repeats and glissandi 154
 repeats and lyrics 275
 repeats and slur 154
 repeats in MIDI 514
 repeats with bar checks 149
 repeats, alternative 157
 repeats, alternative bar numbers 153
 repeats, bar numbers letters 153
 repeats, unfold 157
 repeats, with segno 150
 repeats, with ties 150
 repeats, written-out 157
`\repeatTie` 54, 150, 280
 repetition, using q 166, 336
`\replace` 679
 reserved characters, printing 238
`resetRelativeOctave` 781
 resizing of staves 197
 rest 57, 710
 rest, church 62
 rest, collisions of 64
 rest, condensing ordinary 64
 rest, entering durations 57
 rest, full-measure 60
 rest, invisible 59
 rest, multi-measure 57, 60
 rest, specifying vertical position 57
 rest, whole for a full measure 60
 rest, whole-measure 57
`\rest-by-number` 710
 rest-event 208
 restoring default properties for time signatures 67
`restrainOpenStrings` 336
 rests or multi-measure-rests within text by
 log and dot-count 710
 rests or multi-measure-rests
 within text by string 710
 rests, ancient 432
 rests, splitting 78
 retrogradació, transformació de 14
`\retrograde` 14, 781
 Reverb in MIDI 517
 reversion 119, 732
`\revert` 596
 reverting overrides 596
`\revertTimeSignatureSettings` 67, 781

<code>\rfz</code>	122
<code>rgb color</code>	223
<code>rgb-color</code>	223
<code>\rheel</code>	119, 733
rhythmic staff	186
<code>RhythmicStaff</code>	186
rhythms, showing melody	79
ride bell	733
ride cymbal	733
right aligning text	695
right hand fingerings for fretted instruments	375
<code>\right-align</code>	242, 695
<code>\right-brace</code>	725
<code>\right-column</code>	695
<code>right-margin</code>	527
<code>\rightHandFinger</code>	375, 781
<code>\roman</code>	679
<code>\romanStringNumbers</code>	329
rombe, caps de nota	37
root of chord	405
<code>\rotate</code>	695
rotating objects	617
rotating text	695
<code>\rounded-box</code>	245, 706
<code>\rtoe</code>	119, 733
R	60

S

s	59
<code>\sacredHarpHeads</code>	41
<code>\sacredHarpHeadsMinor</code>	42
<code>\sans</code>	680
<code>SATB</code>	294
scalable vector graphics output	506
<code>\scale</code>	706
<code>\scaleDurations</code>	53, 75, 781
scaling durations	52
scaling markup	706
scaling text	696
Scheme object	739
Scheme variable	738
Scheme, pure containers	629
Scheme, unpure containers	629
<code>\score</code>	464, 468, 711
score inside markup	249
<code>\score-lines</code>	729
<code>score-markup-spacing</code>	525
<code>score-system-spacing</code>	526
<code>scoreTitleMarkup</code>	478
Scottish highland bagpipe	394
script on multi-measure rest	62
scripts	119
seconda volta	147
segno	99, 111, 119, 733
segno on bar line	233
segno, with repeats	150
selecting font size (notation)	215
<code>self-alignment-X</code>	542
Semai form	460
semi-bemoll	9
semi-bemolls	6
Semi-flat symbol appearance	458
semi-sostingut	9
semi-sostinguts	6

semicirculus	733
<code>\semiflat</code>	712
<code>\semiGermanChords</code>	414
<code>\semisharp</code>	712
sense restabliment, estil d'alteracions	33
separate text	236
sesqui-bemoll	9
sesqui-sostingut	9
<code>\sesquiflat</code>	712
<code>\sesquisharp</code>	712
<code>\set</code>	84, 593, 599
<code>set-global-staff-size</code>	533
<code>set-octavation</code>	24
setting extent of text objects	728
setting horizontal text alignment	688
setting subscript in standard font size	677
setting superscript in standard font size	678
<code>settingsFrom</code>	781
seventh chords	405
<code>\sf</code>	122
<code>\sff</code>	122
<code>\sfz</code>	122
<code>\shape</code>	624, 781
shaping slurs and ties	624
shared properties	737
<code>\sharp</code>	712
shift note	173
shift rest, automatic	173
<code>shiftDurations</code>	781
shifting voices	173
<code>\shiftOff</code>	173
<code>\shiftOn</code>	173
<code>\shiftOnn</code>	173
<code>\shiftOnnn</code>	173
<code>short-indent</code>	203, 528
<code>\shortfermata</code>	119, 732
<code>show-available-fonts</code>	253
<code>showFirstLength</code>	505, 738
<code>\showKeySignature</code>	394
<code>showLastLength</code>	505, 738
<code>\showStaffSwitch</code>	322
sidestick	733
signatures, polymetric	75
<code>\signumcongruentiae</code>	119, 733
<code>\simple</code>	680
simple text strings	680
simple text strings with tie characters	712
simplificada, notació	39
simultànies, notes i alteracions	34
simultaneous phrasing slurs	133
simultaneous slurs	130
singer name	287
<code>single</code>	781
single-staff polyphony	169
skip	59, 280, 781
skipping notes in lyrics	280
<code>skipTypesetting</code>	505
<code>slashChordSeparator</code>	413
slashed digits	726
<code>\slashed-digit</code>	726
<code>\slashedGrace</code>	112, 781
slides in tablature notation	345
slur and repeats	154
slur style	131
slur, dashed	131

slur, dashed phrasing	133	spacingTweaks	781
slur, defining dash patterns	131	Span_stem_engraver	323
slur, defining dash patterns for phrasing	134	spanners, modifying	627
slur, dotted	131	special arpeggio symbols	142
slur, dotted phrasing	133	special characters	502
slur, half dashed and half solid	131	special characters in markup mode	238
slur, half solid and half dashed phrasing	133	splash cymbal	733
slur, multiple phrasing	133	splice into tagged music	501
slur, phrasing	130, 133	splitting notes	78
slur, phrasing, defining dash patterns	134	splitting rests	78
slur, simultaneous phrasing	133	\spp	122
slur, solid	131	Sprechgesang	303
slur-event	208	Square neumes ligatures	439
\slurDashed	131	\staccatissimo	119, 120, 732
\slurDashPattern	131, 781	\staccato	119, 120, 732
\slurDotted	131	stacking text in a column	684
\slurDown	130	staff change line	322
\slurHalfDashed	131	staff changes, automatic	320
\slurHalfSolid	131	staff changes, manual	319
\slurNeutral	130	staff distance	541
slurs	130	staff group	187
slurs, above notes	130	staff initiation	186
slurs, below notes	130	staff instantiation	186
slurs, manual placement	130	staff lines, modifying	194
slurs, modifying	624	staff lines, stopping and starting	194
slurs, multiple	130	staff size, setting	533
slurs, simultaneous	130	staff switching	322
\slurSolid	131	staff symbol	194
\slurUp	131	staff symbol, setting of	605
small	215, 241, 680	staff, choir	187
\smallCaps	680	staff, drum	186
\smaller	239, 241, 680	staff, empty	200
smaller notes	209	staff, Frenched	197
smob	739	staff, grand	187
snap pizzicato	331	staff, hiding	200
\snappizzicato	119, 733	staff, multiple	187
snare	733	staff, nested	191
Sol, clau de	17	staff, new	186
Solesmes	426	staff, percussion	186
solid slur	131	staff, piano	187
solo part	178	staff, resizing of	197
soprano, clau de	17	staff, single	186
sos.	325	staff-affinity	542
sostenuto pedal	325	staff-change line	322
\sostenutoOff	325	staff-staff-spacing	542
\sostenutoOn	325	Staff.midiInstrument	518
sostingut	5	Staff_symbol_engraver	200
sostingut, doble	5	staffgroup-staff-spacing	542
Sound	507	standalone text	236
Southern Harmony, caps de nota amb forma	41	standard font size (notation)	218
\southernHarmonyHeads	41	stanza number	286
\southernHarmonyHeadsMinor	42	start of system	187
\sp	122	start repeat	155
space between staves	541	start-repeat	155
space inside systems	541	\startGroup	228
spacer note	59	\startStaff	194, 197
spacer rest	59	\startTrillSpan	145
spaces in lyrics	256	staves, keyboard instruments	318
spaces, in lyrics	263	staves, keyed instruments	318
spacing	557	staves, multiple	187
Spacing lyrics	273	staves, nested	191
spacing section, new	558	staves, piano	318
spacing, display of layout	568	\stdBass	717
spacing, horizontal	556	\stdBassIV	718
spacing, vertical	541	\stdBassV	719

<code>\stdBassVI</code>	720
<code>Stem</code>	323
<code>stem</code>	224
<code>stem, direction</code>	224
<code>stem, down</code>	224
<code>stem, invisible</code>	224
<code>stem, neutral</code>	224
<code>stem, up</code>	224
<code>stem, with slash</code>	114
<code>stem-spacing-correction</code>	557
<code>\stemDown</code>	224
<code>stemLeftBeamCount</code>	94
<code>\stemNeutral</code>	224
<code>stemRightBeamCount</code>	94
<code>stems, cross-staff</code>	323
<code>\stemUp</code>	224
<code>\stencil</code>	726, 739
<code>stencil, removing</code>	610
<code>Stereo balance in MIDI</code>	517
<code>\stopGroup</code>	228
<code>stopped</code>	119, 733
<code>\stopStaff</code>	194, 197, 200
<code>\stopTrillSpan</code>	145
<code>\storePredefinedDiagram</code>	366, 781
<code>string numbers</code>	329, 332
<code>string vs. fingering numbers</code>	332
<code>string, indicating open</code>	329
<code>strings, orchestral</code>	328
<code>strings, writing for</code>	328
<code>\stringTuning</code>	349, 782
<code>stringTunings</code>	348, 362
<code>\strophia</code>	439, 446
<code>strumming rhythms, showing</code>	80
<code>\strut</code>	726
<code>style, rehearsal mark</code>	111
<code>style, slur</code>	131
<code>styledNoteHeads</code>	782
<code>styles, voice</code>	173
<code>\sub</code>	240, 681
<code>subscript</code>	240
<code>subscript text</code>	681
<code>suggestAccidentals</code>	433
<code>\super</code>	240, 681
<code>superscript</code>	240
<code>superscript text</code>	681
<code>sus</code>	408
<code>sustain pedal</code>	325
<code>sustain pedal style</code>	325
<code>\sustainOff</code>	325
<code>\sustainOn</code>	325
<code>SVG output</code>	506
<code>switching fonts</code>	239
<code>syllable durations, automatic</code>	259
<code>symbols, non-musical</code>	247
<code>syntax, markup</code>	237
<code>system</code>	187
<code>system separator mark</code>	193
<code>system start delimiters</code>	187
<code>system start delimiters, nested</code>	191
<code>system-count</code>	529
<code>system-separator-markup</code>	531
<code>system-system-spacing</code>	526
<code>systems-per-page</code>	529

T

<code>tab clef</code>	351
<code>tab micro-tones</code>	351
<code>tab quarter-tones</code>	351
<code>\tabChordRepeats</code>	336, 782
<code>\tabChordRepetition</code>	782
<code>\tabFullNotation</code>	335
<code>tablature</code>	186, 332
<code>tablature and harmonic indications</code>	339
<code>tablature and slides</code>	345
<code>tablature, banjo</code>	332, 348, 380
<code>tablature, bass</code>	348
<code>tablature, bass guitar</code>	348
<code>tablature, cello</code>	348
<code>tablature, custom string tunings</code>	349
<code>tablature, double bass</code>	348
<code>tablature, guitar</code>	332, 348
<code>tablature, lute</code>	380
<code>tablature, mandolin</code>	348
<code>tablature, predefined string tunings</code>	348
<code>tablature, ukulele</code>	348
<code>tablature, viola</code>	348
<code>tablature, violin</code>	348
<code>tablaturs, basic</code>	334
<code>tablaturs, custom</code>	348
<code>tablaturs, default</code>	334
<code>\table</code>	729
<code>\table-of-contents</code>	494, 730
<code>TabStaff</code>	186, 334
<code>tabstaff</code>	186
<code>TabVoice</code>	334
<code>\tag</code>	498, 782
<code>tag groups</code>	501
<code>\tagGroup</code>	501, 782
<code>tam tam</code>	733
<code>tambourine</code>	733
<code>\taor</code>	394
<code>taqasim</code>	460
<code>teaching</code>	33
<code>teaching (enseñanza), estilo de alteraciones</code>	33
<code>teeny</code>	215, 241, 681
<code>Template Arabic music</code>	461
<code>tempo</code>	69
<code>temporary</code>	782
<code>tenor, clau de</code>	17
<code>tenor, clau de, coral</code>	17
<code>\tenuto</code>	119, 120, 732
<code>tessitura</code>	35
<code>text</code>	325, 682
<code>text alignment commands</code>	245
<code>text columns, left-aligned</code>	692
<code>text columns, right-aligned</code>	695
<code>text in columns</code>	244
<code>text in volta bracket</code>	156
<code>text items, non-empty</code>	230
<code>text marks</code>	233
<code>text markup</code>	237
<code>text on bar line</code>	233
<code>text on multi-measure rest</code>	62
<code>text outside margin</code>	231
<code>text padding</code>	246
<code>Text scripts</code>	230
<code>text size</code>	239
<code>text spanners</code>	231

text spanners, formatting	231	timing (within the score)	118
text spread over multiple pages	250	timing information and repeats	154
text, aligning	242	tiny	215, 241, 682
text, centering on the page	244	titles	470
text, decorating	245	\tocItem	494, 782
text, framing	245	tom tom	733
text, horizontal alignment	242	top-level text	236
text, justified	244	top-margin	524
text, keeping inside margin	231	top-markup-spacing	526
text, multi-line	244	top-system-spacing	526
Text, other languages	230	toplevel-bookparts	738
text, separate	236	toplevel-scores	738
text, standalone	236	transcription of mensural music	190
text, top-level	236	transformació de retrogradació	14
text, vertical alignment	242	transformacions modals	14
text, wordwrapped	244	\translate	243, 696
\textLengthOff	62, 231	\translate-scaled	243, 696
\textLengthOn	62, 231	translating text	696
\textSpannerDown	232	\transparent	726
\textSpannerNeutral	232	transparent notes	221
\textSpannerUp	232	transparent, making objects	611
Thorough bass	417	transport	10, 15
\thumb	119, 219, 732	transport modal	15
thumb marking	119	\transpose	5, 10, 13, 782
thumb-script	219	transposed clefs, visibility of	616
tick mark	135	\transposedCueDuring	212, 782
tie	53, 682	transposició	10
tie-ing text	682	transposició d'altures	10
\tied-lyric	712	transposició de MIDI	26
\tieDashed	55	transposició de notes	10
tieDashPattern	782	transposició i mode relatiu	5
\tieDotted	55	transposició instrumental	26
\tieDown	55	transposició modal	15
\tieNeutral	55	transposició, de octava, opcional	17
ties and chords	54	transposing fret diagrams	364
ties and volta brackets	54	\transposition	26, 206, 782
ties, alternative endings	150	transpositor, instrument	26
ties, appearance	55	transpositores, claus	17
ties, dashed	55	transpositors, instruments	11
ties, dotted	55	tre corde	325
ties, in lyrics	263	\treCorde	325
ties, in repeats	150	tremolo	162
ties, laissez vibrer	55	tremolo beams	162
ties, modifying	624	tremolo marks	162
ties, placement	55	tremolo, cross-staff	163
ties, repeating	54	triads	405
\tieSolid	55	\triangle	247, 707, 733
\tieUp	55	trill	119, 145, 732
timbale	733	trill with accidental	146
\time	65, 84, 782	trills	145
time administration	118	trills, pitched	146
time signature	65	triplet formatting	49
time signature default settings	66	triplets	48
time signature properties,		tuning, non-Western	456
restoring default values	67	tunings, banjo	380
time signature style	65, 430	tunings, lute	381
time signature visibility	65	\tuplet	48, 75, 783
time signature, compound	77	tuplet bracket placement	48
time signature, double	75	tuplet formatting	49
time signature, mensural	430	tuplet grouping	48
time signature, mid-measure	72	Tuplet number changes	49
time signature, multiple	580	\tupletDown	48
time signature, polymetric	75	\tupletNeutral	48
times	782	TupletNumber	49
timeSignatureFraction	75	tupletNumberFormatFunction	49

tuplets	48
tupletSpan	783
tupletSpannerDuration	49
\tupletUp	48
Turkish music	462
Turkish note names	462
turn	119, 732
\tweak	597, 599, 783
tweak, relation to \override	599
tweaking	597
tweaking control points	599
tweaking grace notes	114
two-sided	528
\type	586
typeface	737
typeset text	237
\typewriter	683

U

U.C.	325
ukulele	353
una corda	325
\unaCorda	325
\underline	239, 683
underlining text	683
\undertie	683
undertie-ing text	683
undo	783
unfold	157
unfold repeat	157
unfold repeat, alternate endings	157
\unfoldRepeats	514, 783
\unHideNotes	221
Unicode	503
unmetered music	73, 118
unmetered music, accidentals	73
unmetered music, bar lines	73
unmetered music, bar numbers	73
unmetered music, beams	73
unmetered music, line breaks	75
unmetered music, page breaks	75
unpure containers, Scheme	629
\unset	594
up bow indication	329
upbeat	72
upbeat in a repeat	149
upbow	119, 329, 733
\upmordent	119, 732
\upprall	119, 732
\upright	683
UTF-8	502

V

varcoda	119, 733
variables	469
variables, use of	496
Vaticana, Editio	425, 426
VaticanaStaff	186, 435
VaticanaVoice	435
\vcenter	696
\verbatim-file	726
vertical lines between staves	226
vertical positioning of dynamics	124

vertical spacing	541, 561
vertical text alignment	242
VerticalAxisGroup	542
vertically centering text	696
\verylongfermata	119, 732
veu, estil d'alteracions	29
veus, alteracions en diverses	30
vibraslap	733
violí, clau de	17
\virga	439, 446
\virgula	438
visibility of objects	610
visibility of transposed clefs	616
Voice	169
voice	28, 29, 169
voice styles	173
voice, following	322
\voiceFourStyle	173
\voiceNeutralStyle	173
\voiceOne	169
\voiceOne ... \voiceFour	169
\voiceOneStyle	173
voices, divided	296
voices, multiple	173
voices, \partcombine with \autoBeamOff	83
voices, quoting	206, 209
\voiceThreeStyle	173
\voiceTwoStyle	173
\void	519, 783
volta	147
volta bracket	155
volta bracket with text	156
volta brackets and ties	54
volta, prima	147
volta, seconda	147
\vspace	696

W

Walker, caps de nota amb forma	41
\walkerHeads	41
\walkerHeadsMinor	42
whichBar	103
whistle	733
White mensural ligatures	434
\whiteout	727
whitespace	469
whole rest for a full measure	60
wind instruments	391
\with	579, 584
with-color	222, 727
\with-dimensions	728
\with-dimensions-from	728
\with-link	728
\with-url	707
withMusicProperty	783
woodblock	733
\woodwind-diagram	716
\wordwrap	244, 698
\wordwrap-field	697
\wordwrap-internal	730
\wordwrap-lines	250, 730
\wordwrap-string	698
\wordwrap-string-internal	730

wordwrapped text 244
writing music in parallel 183
written-out repeats 157

X

x11 color 222, 223
x11-color 222, 223
X-offset 542
xNote 783
xNotesOn 783