This manual provides a reference for all notation that can be produced with LilyPond version 2.18.2. It assumes that the reader is familiar with the material in the Section “Learning Manual” in Learning Manual.

For more information about how this manual fits with the other documentation, or to read this manual in other formats, see Section “Manuals” in General Information. If you are missing any manuals, the complete documentation can be found at http://www.lilypond.org/.

Copyright © 1999–2012 by the authors.

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”.

For LilyPond version 2.18.2
# Table of Contents

## 1 Musical notation

1.1 Pitches ......................................................... 1
   1.1.1 Writing pitches ........................................... 1
       Absolute octave entry .................................... 1
       Relative octave entry .................................... 2
       Accidentals .................................................. 5
       Note names in other languages ............................. 7
   1.1.2 Changing multiple pitches ................................. 9
       Octave checks ............................................... 9
       Transpose .................................................... 10
       Inversion ..................................................... 13
       Retrograde ................................................... 13
       Modal transformations ....................................... 14
   1.1.3 Displaying pitches ...................................... 16
       Clef .......................................................... 16
       Key signature ................................................ 20
       Ottava brackets ............................................. 22
       Instrument transpositions .................................. 24
       Automatic accidentals ...................................... 25
       Ambitus ....................................................... 32
   1.1.4 Note heads ............................................... 34
       Special note heads .......................................... 34
       Easy notation note heads ................................... 36
       Shape note heads ............................................. 37
       Improvisation ................................................ 40

1.2 Rhythms ....................................................... 41
   1.2.1 Writing rhythms ......................................... 41
       Durations ..................................................... 41
       Tuplets ....................................................... 44
       Scaling durations ......................................... 48
       Ties .......................................................... 49
   1.2.2 Writing rests ............................................ 52
       Rests ........................................................ 52
       Invisible rests ............................................... 54
       Full measure rests ......................................... 55
   1.2.3 Displaying rhythms ...................................... 59
       Time signature ............................................... 59
       Metronome marks ............................................. 64
       Upbeats ....................................................... 67
       Unmetered music ............................................. 68
       Polymetric notation ........................................ 69
       Automatic note splitting ................................... 72
       Showing melody rhythms ................................... 73
   1.2.4 Beams ...................................................... 75
       Automatic beams ............................................. 76
       Setting automatic beam behavior ......................... 78
       Manual beams ............................................... 86
       Feathered beams ............................................. 89
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2.5 Bars</td>
<td>90</td>
</tr>
<tr>
<td>Bar lines</td>
<td>90</td>
</tr>
<tr>
<td>Bar numbers</td>
<td>96</td>
</tr>
<tr>
<td>Bar and bar number checks</td>
<td>101</td>
</tr>
<tr>
<td>Rehearsal marks</td>
<td>101</td>
</tr>
<tr>
<td>1.2.6 Special rhythmic concerns</td>
<td>103</td>
</tr>
<tr>
<td>Grace notes</td>
<td>104</td>
</tr>
<tr>
<td>Aligning to cadenzas</td>
<td>109</td>
</tr>
<tr>
<td>Time administration</td>
<td>110</td>
</tr>
<tr>
<td>1.3 Expressive marks</td>
<td>111</td>
</tr>
<tr>
<td>1.3.1 Expressive marks attached to notes</td>
<td>111</td>
</tr>
<tr>
<td>Articulations and ornamentations</td>
<td>111</td>
</tr>
<tr>
<td>Dynamics</td>
<td>114</td>
</tr>
<tr>
<td>New dynamic marks</td>
<td>119</td>
</tr>
<tr>
<td>1.3.2 Expressive marks as curves</td>
<td>122</td>
</tr>
<tr>
<td>Slurs</td>
<td>122</td>
</tr>
<tr>
<td>Phrasing slurs</td>
<td>124</td>
</tr>
<tr>
<td>Breath marks</td>
<td>126</td>
</tr>
<tr>
<td>Falls and doits</td>
<td>127</td>
</tr>
<tr>
<td>1.3.3 Expressive marks as lines</td>
<td>128</td>
</tr>
<tr>
<td>Glissando</td>
<td>128</td>
</tr>
<tr>
<td>Arpeggio</td>
<td>133</td>
</tr>
<tr>
<td>Trills</td>
<td>135</td>
</tr>
<tr>
<td>1.4 Repeats</td>
<td>137</td>
</tr>
<tr>
<td>1.4.1 Long repeats</td>
<td>138</td>
</tr>
<tr>
<td>Normal repeats</td>
<td>138</td>
</tr>
<tr>
<td>Manual repeat marks</td>
<td>145</td>
</tr>
<tr>
<td>Written-out repeats</td>
<td>148</td>
</tr>
<tr>
<td>1.4.2 Short repeats</td>
<td>150</td>
</tr>
<tr>
<td>Percent repeats</td>
<td>150</td>
</tr>
<tr>
<td>Tremolo repeats</td>
<td>152</td>
</tr>
<tr>
<td>1.5 Simultaneous notes</td>
<td>153</td>
</tr>
<tr>
<td>1.5.1 Single voice</td>
<td>154</td>
</tr>
<tr>
<td>Chorded notes</td>
<td>154</td>
</tr>
<tr>
<td>Chord repetition</td>
<td>156</td>
</tr>
<tr>
<td>Simultaneous expressions</td>
<td>157</td>
</tr>
<tr>
<td>Clusters</td>
<td>158</td>
</tr>
<tr>
<td>1.5.2 Multiple voices</td>
<td>159</td>
</tr>
<tr>
<td>Single-staff polyphony</td>
<td>159</td>
</tr>
<tr>
<td>Voice styles</td>
<td>162</td>
</tr>
<tr>
<td>Collision resolution</td>
<td>162</td>
</tr>
<tr>
<td>Automatic part combining</td>
<td>167</td>
</tr>
<tr>
<td>Writing music in parallel</td>
<td>172</td>
</tr>
<tr>
<td>1.6 Staff notation</td>
<td>174</td>
</tr>
<tr>
<td>1.6.1 Displaying staves</td>
<td>175</td>
</tr>
<tr>
<td>Instantiating new staves</td>
<td>175</td>
</tr>
<tr>
<td>Grouping staves</td>
<td>176</td>
</tr>
<tr>
<td>Nested staff groups</td>
<td>179</td>
</tr>
<tr>
<td>Separating systems</td>
<td>181</td>
</tr>
<tr>
<td>1.6.2 Modifying single staves</td>
<td>182</td>
</tr>
<tr>
<td>Staff symbol</td>
<td>182</td>
</tr>
<tr>
<td>Ossia staves</td>
<td>185</td>
</tr>
<tr>
<td>Hiding staves</td>
<td>189</td>
</tr>
<tr>
<td>1.6.3 Writing parts</td>
<td>192</td>
</tr>
</tbody>
</table>
2.4.3 Banjo .................................................. 360
Banjo tablatures ........................................ 361
2.5 Percussion ............................................. 361
2.5.1 Common notation for percussion ...
References for percussion ............................... 361
Basic percussion notation ............................... 362
Drum rolls ................................................ 363
Pitched percussion ....................................... 363
Percussion staves ........................................ 363
Custom percussion staves .............................. 365
Ghost notes ................................................ 369
2.6 Wind instruments ...................................... 370
2.6.1 Common notation for wind instruments ...
References for wind instruments ...................... 370
Fingerings ............................................... 371
2.6.2 Bagpipes ............................................ 373
Bagpipe definitions ..................................... 373
Bagpipe example ........................................ 374
2.6.3 Woodwinds ......................................... 375
2.6.3.1 Woodwind diagrams .......................... 375
2.7 Chord notation ......................................... 384
2.7.1 Chord mode ......................................... 384
Chord mode overview ................................... 384
Common chords .......................................... 385
Extended and altered chords ......................... 387
2.7.2 Displaying chords ................................. 389
Printing chord names .................................. 389
Customizing chord names ............................. 392
2.7.3 Figured bass ......................................... 397
Introduction to figured bass ......................... 398
Entering figured bass .................................. 398
Displaying figured bass ............................... 401
2.8 Contemporary music .................................. 403
2.8.1 Pitch and harmony in contemporary music ...
References for pitch and harmony in contemporary music .......................... 403
Microtonal notation .................................... 403
Contemporary key signatures and harmony ......... 404
2.8.2 Contemporary approaches to rhythm ............ 404
References for contemporary approaches to rhythm ................................ 404
Tuplets in contemporary music ....................... 404
Contemporary time signatures ....................... 404
Extended polymetric notation ....................... 404
Beams in contemporary music ....................... 404
Bar lines in contemporary music .................... 404
2.8.3 Graphical notation ................................ 404
2.8.4 Contemporary scoring techniques .............. 404
2.8.5 New instrumental techniques .................... 404
2.8.6 Further reading and scores of interest ......... 404
Books and articles on contemporary musical notation .......................... 404
Scores and musical examples ......................... 404
2.9 Ancient notation ...................................... 404
2.10 World music

2.10.1 Common notation for non-Western music

Extending notation and tuning systems

2.10.2 Arabic music

References for Arabic music

Arabic note names

Arabic key signatures

Arabic time signatures

Arabic music example

Further reading for Arabic music

2.10.3 Turkish classical music

References for Turkish classical music

Turkish note names

2.9.3 Typesetting mensural music

Mensural contexts

Mensural clefs

Mensural time signatures

Mensural note heads

Mensural flags

Mensural rests

Mensural accidentals and key signatures

Annotational accidentals (musica ficta)

White mensural ligatures

2.9.4 Typesetting Gregorian chant

Gregorian chant contexts

Gregorian clefs

Gregorian accidentals and key signatures

Divisiones

Gregorian articulation signs

Augmentum dots (more)

Gregorian square neume ligatures

2.9.5 Typesetting Kievan square notation

Kievan contexts

Kievan clefs

Kievan notes

Kievan accidentals

Kievan bar line

Kievan melismata

2.9.6 Working with ancient music—scenarios and solutions

Incipits

Mensurstriche layout

Transcribing Gregorian chant

Ancient and modern from one source

Editorial markings

2.10 World music

2.10.1 Common notation for non-Western music

Extending notation and tuning systems

2.10.2 Arabic music

References for Arabic music

Arabic note names

Arabic key signatures

Arabic time signatures

Arabic music example

Further reading for Arabic music

2.10.3 Turkish classical music

References for Turkish classical music

Turkish note names
3 General input and output .................................................. 442
3.1 Input structure ..................................................................... 442
  3.1.1 Structure of a score ......................................................... 442
  3.1.2 Multiple scores in a book ................................................. 443
  3.1.3 Multiple output files from one input file ......................... 444
  3.1.4 Output file names ......................................................... 445
  3.1.5 File structure .......................................................... 446
3.2 Titles and headers .......................................................... 448
  3.2.1 Creating titles headers and footers .................................. 448
    Titles explained ............................................................ 448
    Default layout of bookpart and score titles ....................... 451
    Default layout of headers and footers ................................ 454
  3.2.2 Custom titles headers and footers .................................. 455
    Custom text formatting for titles ...................................... 455
    Custom layout for titles ................................................. 456
    Custom layout for headers and footers ............................. 459
  3.2.3 Creating footnotes ...................................................... 460
    Footnotes in music expressions ........................................ 460
    Footnotes in stand-alone text .......................................... 466
  3.2.4 Reference to page numbers ........................................... 469
  3.2.5 Table of contents ...................................................... 470
3.3 Working with input files ................................................. 472
  3.3.1 Including LilyPond files .............................................. 472
  3.3.2 Different editions from one source .................................. 473
    Using variables ............................................................ 474
    Using tags .................................................................. 475
    Using global settings .................................................... 478
  3.3.3 Special characters ...................................................... 478
    Text encoding ............................................................. 478
    Unicode .................................................................... 479
    ASCII aliases .............................................................. 480
3.4 Controlling output .......................................................... 481
  3.4.1 Extracting fragments of music ....................................... 481
  3.4.2 Skipping corrected music .............................................. 481
  3.4.3 Alternative output formats .......................................... 482
  3.4.4 Replacing the notation font ......................................... 482
3.5 MIDI output ................................................................. 483
  3.5.1 Creating MIDI files ...................................................... 483
  3.5.2 MIDI Instruments ........................................................ 486
  3.5.3 What goes into the MIDI output? .................................... 486
    Supported in MIDI ........................................................ 486
    Unsupported in MIDI ...................................................... 486
  3.5.4 Repeats in MIDI ........................................................ 487
  3.5.5 Controlling MIDI dynamics .......................................... 487
    Dynamic marks ............................................................ 488
    Overall MIDI volume ..................................................... 488
    Equalizing different instruments (i) .................................. 489
    Equalizing different instruments (ii) ................................. 490
  3.5.6 Percussion in MIDI ...................................................... 491
  3.5.7 The Articulate script ................................................... 492
3.6 Extracting musical information ......................................... 492
  3.6.1 Displaying LilyPond notation ........................................ 492
  3.6.2 Displaying scheme music expressions ............................ 493
  3.6.3 Saving music events to a file ....................................... 493
4 Spacing issues ................................................................. 494

4.1 Page layout ................................................................. 494
  4.1.1 The \paper block .................................................. 494
  4.1.2 Paper size and automatic scaling ............................... 495
    Setting the paper size .............................................. 495
    Automatic scaling to paper size .................................. 496
  4.1.3 Fixed vertical spacing \paper variables ....................... 496
  4.1.4 Flexible vertical spacing \paper variables ................... 497
    Structure of flexible vertical spacing alists .................. 497
    List of flexible vertical spacing \paper variables .......... 498
  4.1.5 Horizontal spacing \paper variables ......................... 499
    \paper variables for widths and margins ...................... 499
    \paper variables for two-sided mode ............................ 500
    \paper variables for shifts and indents ...................... 501
  4.1.6 Other \paper variables ........................................ 501
    \paper variables for line breaking .............................. 501
    \paper variables for page breaking ............................. 502
    \paper variables for page numbering ........................... 503
    Miscellaneous \paper variables ................................. 503

4.2 Score layout ............................................................ 504
  4.2.1 The \layout block ............................................... 504
  4.2.2 Setting the staff size ......................................... 506

4.3 Breaks ........................................................................ 507
  4.3.1 Line breaking ..................................................... 507
  4.3.2 Page breaking ..................................................... 509
  4.3.3 Optimal page breaking .......................................... 510
  4.3.4 Optimal page turning ........................................... 510
  4.3.5 Minimal page breaking ......................................... 511
  4.3.6 One-line page breaking ....................................... 511
  4.3.7 Explicit breaks ................................................. 511
  4.3.8 Using an extra voice for breaks .............................. 513

4.4 Vertical spacing ........................................................... 515
  4.4.1 Flexible vertical spacing within systems .................... 515
    Within-system spacing properties ................................ 515
    Spacing of ungrouped staves ..................................... 518
    Spacing of grouped staves ........................................ 519
    Spacing of non-staff lines ........................................ 521
  4.4.2 Explicit staff and system positioning ........................ 522
  4.4.3 Vertical collision avoidance ................................... 529

4.5 Horizontal spacing ....................................................... 530
  4.5.1 Horizontal spacing overview ................................... 530
  4.5.2 New spacing area ................................................ 532
  4.5.3 Changing horizontal spacing ................................... 532
  4.5.4 Line length ....................................................... 534
  4.5.5 Proportional notation .......................................... 535

4.6 Fitting music onto fewer pages ........................................ 541
  4.6.1 Displaying spacing .............................................. 541
  4.6.2 Changing spacing ................................................. 542
5 Changing defaults ................................................................. 545
  5.1 Interpretation contexts ...................................................... 545
    5.1.1 Contexts explained .................................................. 545
      Output definitions - blueprints for contexts ........................ 545
      Score - the master of all contexts .................................. 546
      Top-level contexts - staff containers ............................... 546
      Intermediate-level contexts - staves ............................... 546
      Bottom-level contexts - voices ..................................... 547
  5.1.2 Creating and referencing contexts ................................ 547
  5.1.3 Keeping contexts alive ................................................ 550
  5.1.4 Modifying context plug-ins ......................................... 553
    5.1.5 Changing context default settings ............................... 555
      Changing all contexts of the same type ............................ 555
      Changing just one specific context ................................ 558
    Order of precedence .................................................... 559
  5.1.6 Defining new contexts ................................................ 560
  5.1.7 Context layout order .................................................. 562
  5.2 Explaining the Internals Reference ................................... 564
    5.2.1 Navigating the program reference ................................ 564
    5.2.2 Layout interfaces .................................................. 564
    5.2.3 Determining the grob property ................................... 565
    5.2.4 Naming conventions ................................................ 566
  5.3 Modifying properties ..................................................... 567
    5.3.1 Overview of modifying properties ................................. 567
    5.3.2 The \set command .................................................. 567
    5.3.3 The \override command .......................................... 569
    5.3.4 The \tweak command ............................................... 571
    5.3.5 \set vs. \override ............................................... 573
    5.3.6 Modifying alists .................................................. 573
  5.4 Useful concepts and properties ........................................ 575
    5.4.1 Input modes .......................................................... 575
    5.4.2 Direction and placement ........................................... 577
      Articulation direction indicators .................................. 577
      The direction property .............................................. 577
    5.4.3 Distances and measurements ....................................... 578
    5.4.4 Staff symbol properties .......................................... 578
    5.4.5 Spanners ............................................................. 579
      Using the spanner-interface ......................................... 579
      Using the line-spanner-interface ................................... 581
    5.4.6 Visibility of objects ............................................... 584
      Removing the stencil .................................................. 584
      Making objects transparent .......................................... 584
      Painting objects white ............................................... 585
      Using break-visibility ............................................... 585
      Special considerations .............................................. 587
    5.4.7 Line styles .......................................................... 590
    5.4.8 Rotating objects ................................................... 590
      Rotating layout objects ............................................... 590
      Rotating markup ...................................................... 591
  5.5 Advanced tweaks .......................................................... 591
    5.5.1 Aligning objects .................................................... 591
      Setting X-offset and Y-offset directly ............................. 592
      Using the side-position-interface .................................. 592
      Using the self-alignment-interface ................................ 593
Using the break-alignable-interface ........................................... 594
5.5.2 Vertical grouping of grobs .................................................. 596
5.5.3 Modifying stencils ............................................................ 596
5.5.4 Modifying shapes .............................................................. 597
      Modifying ties and slurs .................................................... 597
5.5.5 Modifying broken spanners ................................................ 601
Using \alterBroken ................................................................. 601
5.5.6 Unpure-pure containers ..................................................... 602
5.6 Using music functions ......................................................... 604
5.6.1 Substitution function syntax .............................................. 604
5.6.2 Substitution function examples .......................................... 605

Appendix A  Notation manual tables ........................................... 607
A.1 Chord name chart ............................................................... 607
A.2 Common chord modifiers ...................................................... 608
A.3 Predefined string tunings ..................................................... 611
A.4 Predefined fretboard diagrams .............................................. 613
      Diagrams for Guitar .......................................................... 613
      Diagrams for Ukulele .......................................................... 615
      Diagrams for Mandolin ....................................................... 616
A.5 Predefined paper sizes ....................................................... 618
A.6 MIDI instruments ............................................................... 622
A.7 List of colors ................................................................. 622
A.8 The Feta font ................................................................. 624
      Clef glyphs ................................................................. 624
      Time Signature glyphs ..................................................... 624
      Number glyphs ............................................................. 625
      Accidental glyphs ........................................................ 625
      Default Notehead glyphs .................................................. 626
      Special Notehead glyphs .................................................. 626
      Shape-note Notehead glyphs ............................................. 627
      Rest glyphs ............................................................... 631
      Flag glyphs ............................................................... 632
      Dot glyphs ............................................................... 632
      Dynamic glyphs .......................................................... 632
      Script glyphs ............................................................. 633
      Arrowhead glyphs ......................................................... 635
      Bracket-tip glyphs ....................................................... 635
      Pedal glyphs ............................................................... 635
      Accordion glyphs .......................................................... 636
      Tie glyphs ............................................................... 636
      Vaticana glyphs ............................................................ 636
      Medicae glyphs ............................................................ 637
      Hufnagel glyphs ............................................................ 638
      Mensural glyphs ............................................................ 638
      Neomensural glyphs ....................................................... 642
      Petrucci glyphs ............................................................ 643
      Solesmes glyphs ........................................................... 644
      Kievan Notation glyphs .................................................... 644
A.9 Note head styles .............................................................. 645
A.10 Text markup commands ..................................................... 645
A.10.1 Font ................................................................. 645
A.10.2 Align ............................................................... 654
A.10.3 Graphic ............................................................... 669
1 Musical notation

This chapter explains how to create musical notation.

1.1 Pitches

This section discusses how to specify the pitch of notes. There are three steps to this process: input, modification, and output.

1.1.1 Writing pitches

This section discusses how to input pitches. There are two different ways to place notes in octaves: absolute and relative mode. In most cases, relative mode will be more convenient.

Absolute octave entry

A pitch name is specified using lowercase letters a through g. The note names c to b are engraved in the octave below middle C.

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

Other octaves may be specified with a single quote (’) or comma (,) character. Each ’ raises the pitch by one octave; each , lowers the pitch by an octave.
Music can be marked explicitly as being in absolute octave notation by preceding it with \absolute:
\absolute musicexpr

will be interpreted in absolute octave entry mode regardless of the context it appears in.

See also
Snippets: Section “Pitches” in Snippets.

Relative octave entry
Absolute octave entry requires specifying the octave for every single note. Relative octave entry, in contrast, specifies each octave in relation to the last note: changing one note’s octave will affect all of the following notes.

Relative note mode has to be entered explicitly using the \relative command:
\relative { musicexpr

In relative mode, each note is assumed to be as close to the previous note as possible. This means that the octave of each pitch inside musicexpr is calculated as follows:

- If no octave changing mark is used on a pitch, its octave is calculated so that the interval with the previous note is less than a fifth. This interval is determined without considering accidentals.
- An octave changing mark ' or , can be added to respectively raise or lower a pitch by an extra octave, relative to the pitch calculated without an octave mark.
- Multiple octave changing marks can be used. For example, '' and ,, will alter the pitch by two octaves.
- The pitch of the first note is relative to startpitch. startpitch is specified in absolute octave mode. Which choices are meaningful?

an octave of c
Identifying middle C with c' is quite basic, so finding octaves of c tends to be straightforward. If your music starts with gis above c''', you’d write something like \relative c''' { gis' ... }

an octave of the first note inside
Writing \relative gis''' { gis ... } makes it easy to determine the absolute pitch of the first note inside.
no explicit starting pitch

This (namely writing \relative { gis''' ... }) can be viewed as a compact version of the previous option: the first note inside is written in absolute pitch itself. This happens to be equivalent to choosing f as the reference pitch.

The documentation will usually employ the first option.

Here is the relative mode shown in action:
\relative c {
  \clef bass
  c d e f
g a b c
d e f g
}

Octave changing marks are used for intervals greater than a fourth:
\relative c' {
  c g c f,
c' a, e'' c
}

A note sequence without a single octave mark can nevertheless span large intervals:
\relative c {
  c f b e
  a d g c
}

When \relative blocks are nested, the innermost \relative block applies.
\relative c' {
  c d e f
  \relative c'' {
    c d e f
  }
}

\relative has no effect on \chordmode blocks.
Chapter 1: Musical notation

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

-\relative is not allowed inside of \chordmode blocks.

Music inside a \transpose block is absolute unless a \relative is included.

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

-As explained above, the octave of pitches is calculated only with the note names, regardless of any alterations. Therefore, an E-double-sharp following a B will be placed higher, while an F-double-flat will be placed lower. In other words, a double-augmented fourth is considered a smaller interval than a double-diminished fifth, regardless of the number of semitones that each interval contains.
A sharp pitch is made by adding is to the note name, and a flat pitch by adding es. As you might expect, a double sharp or double flat is made by adding isis or eses. This syntax is derived from Dutch note naming conventions. To use other names for accidentals, see [Note names in other languages], page 7.

A natural will cancel the effect of an accidental or key signature. However, naturals are not encoded into the note name syntax with a suffix; a natural pitch is shown as a simple note name:

A natural will cancel the effect of an accidental or key signature. However, naturals are not encoded into the note name syntax with a suffix; a natural pitch is shown as a simple note name:

Quarter tones may be added; the following is a series of Cs with increasing pitches:

Quarter tones may be added; the following is a series of Cs with increasing pitches:
Normally accidentals are printed automatically, but you may also print them manually. A
reminder accidental can be forced by adding an exclamation mark ! after the pitch. A cautionary
accidental (i.e., an accidental within parentheses) can be obtained by adding the question mark ?
after the pitch. These extra accidentals can also be used to produce natural signs.

\cis\cis\cis!\cis?\ccc!\cc?

Accidentals on tied notes are only printed at the beginning of a new system:

\cis1^\cis^\break\cis

Selected Snippets

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'' {\overline \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\texttt{extraNatural} property to\texttt{f} in the\texttt{Staff} context.

\relative c' {aeses4aesaisa \set\Staff.extraNatural = ##f aeses4aesaisa}
Chapter 1: Musical notation

See also


Notation Reference: [Automatic accidentals], page 25, [Annotational accidentals (musica ficta)], page 414, [Note names in other languages], page 7.

Snippets: Section “Pitches” in Snippets.


Known issues and warnings

There are no generally accepted standards for denoting quarter-tone accidentals, so LilyPond’s symbol does not conform to any standard.

Note names in other languages

There are predefined sets of note and accidental names for various other languages. Selecting the note name language is usually done at the beginning of the file; the following example is written using Italian note names:

\language "italiano"

\relative do' { 
   do re mi sib
}

The available languages and the note names they define are:

<table>
<thead>
<tr>
<th>Language</th>
<th>Note Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>nederlands</td>
<td>c d e f g a bes b</td>
</tr>
<tr>
<td>catalan</td>
<td>do re mi fa sol la sib si</td>
</tr>
<tr>
<td>deutsch</td>
<td>c d e f g a b h</td>
</tr>
<tr>
<td>english</td>
<td>c d e f g a bf b</td>
</tr>
<tr>
<td>espanol</td>
<td>or do re mi fa sol la sib si</td>
</tr>
<tr>
<td>espaniol</td>
<td>or do re mi fa sol la sib si</td>
</tr>
<tr>
<td>italiano</td>
<td>or do re mi fa sol la sib si</td>
</tr>
<tr>
<td>francais</td>
<td>do re mi fa sol la sib si</td>
</tr>
<tr>
<td>norsk</td>
<td>c d e f g a b h</td>
</tr>
<tr>
<td>portugues</td>
<td>do re mi fa sol la sib si</td>
</tr>
</tbody>
</table>
### Chapter 1: Musical notation

In addition to note names, accidental suffixes may also vary depending on the language:

<table>
<thead>
<tr>
<th>Language</th>
<th>sharp</th>
<th>flat</th>
<th>double sharp</th>
<th>double flat</th>
</tr>
</thead>
<tbody>
<tr>
<td>nederlands</td>
<td>-is</td>
<td>-es</td>
<td>-isis</td>
<td>-eses</td>
</tr>
<tr>
<td>catalan</td>
<td>-d/-s</td>
<td>-b</td>
<td>-dd/-ss</td>
<td>-bb</td>
</tr>
<tr>
<td>deutsch</td>
<td>-is</td>
<td>-es</td>
<td>-isis</td>
<td>-eses</td>
</tr>
<tr>
<td>english</td>
<td>-s/-sharp</td>
<td>-f/-flat</td>
<td>-ss/-x/-sharpsharp</td>
<td>-ff/-flatflat</td>
</tr>
<tr>
<td>espanol</td>
<td>or -s</td>
<td>-b</td>
<td>-ss/-x</td>
<td>-bb</td>
</tr>
<tr>
<td>italiano</td>
<td>or -d</td>
<td>-b</td>
<td>-dd</td>
<td>-bb</td>
</tr>
<tr>
<td>fransais</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>norsk</td>
<td>-iss/-is</td>
<td>-ess/-es</td>
<td>-ississ/-isis</td>
<td>-essess/-eses</td>
</tr>
<tr>
<td>portugues</td>
<td>-s</td>
<td>-b</td>
<td>-ss</td>
<td>-bb</td>
</tr>
<tr>
<td>suomi</td>
<td>-is</td>
<td>-es</td>
<td>-isis</td>
<td>-eses</td>
</tr>
<tr>
<td>svenska</td>
<td>-iss</td>
<td>-ess</td>
<td>-ississ</td>
<td>-essess</td>
</tr>
<tr>
<td>vlaams</td>
<td>-k</td>
<td>-b</td>
<td>-kk</td>
<td>-bb</td>
</tr>
</tbody>
</table>

In Dutch, aes is contracted to as, but both forms are accepted in LilyPond. Similarly, both es and ees are accepted. This also applies to aeses / ase and eeses / eses. Sometimes only these contracted names are defined in the corresponding language files.

\[
a2 \ as \ e \ es \ a \ ases \ e \ eses
\]

Some music uses microtones whose alterations are fractions of a ‘normal’ sharp or flat. The following table lists note names for quarter-tone accidentals in various languages; here the prefixes semi- and sesqui- respectively mean ‘half’ and ‘one and a half’. Languages that do not appear in this table do not provide special note names yet.

<table>
<thead>
<tr>
<th>Language</th>
<th>semi-sharp</th>
<th>semi-flat</th>
<th>sesqui-sharp</th>
<th>sesqui-flat</th>
</tr>
</thead>
<tbody>
<tr>
<td>nederlands</td>
<td>-ih</td>
<td>-eh</td>
<td>-isih</td>
<td>-eseh</td>
</tr>
<tr>
<td>deutsch</td>
<td>-ih</td>
<td>-eh</td>
<td>-isih</td>
<td>-eseh</td>
</tr>
<tr>
<td>english</td>
<td>-qs</td>
<td>-qf</td>
<td>-tqs</td>
<td>-tqf</td>
</tr>
<tr>
<td>espanol</td>
<td>or -cs</td>
<td>-cb</td>
<td>-tcs</td>
<td>-tcb</td>
</tr>
<tr>
<td>español</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>italiano</td>
<td>or -sd</td>
<td>-sb</td>
<td>-dsd</td>
<td>-bsb</td>
</tr>
<tr>
<td>fransais</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>portugues</td>
<td>-sqt</td>
<td>-bqt</td>
<td>-stqt</td>
<td>-btqt</td>
</tr>
</tbody>
</table>

Most languages presented here are commonly associated with Western classical music, also referred to as Common Practice Period. However, alternate pitches and tuning systems are also supported: see Section 2.10.1 [Common notation for non-Western music], page 434.

**See also**

1.1.2 Changing multiple pitches

This section discusses how to modify pitches.

Octave checks

In relative mode, it is easy to forget an octave changing mark. Octave checks make such errors easier to find by displaying a warning and correcting the octave if a note is found in an unexpected octave.

To check the octave of a note, specify the absolute octave after the \= symbol. This example will generate a warning (and change the pitch) because the second note is the absolute octave d'' instead of d' as indicated by the octave correction.

\relative c'' {
  c2 d=4 d
  e2 f
}

The octave of notes may also be checked with the \octaveCheck controlpitch command. controlpitch is specified in absolute mode. This checks that the interval between the previous note and the controlpitch is within a fourth (i.e., the normal calculation of relative mode). If this check fails, a warning is printed, but the previous note is not changed. Future notes are relative to the controlpitch.

\relative c'' {
  c2 d
  \octaveCheck c'
  e2 f
}

Compare the two bars below. The first and third \octaveCheck checks fail, but the second one does not fail.

\relative c'' {
  c4 f g f

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

A music expression can be transposed with \transpose. The syntax is

\transpose frompitch topitch musicexpr

This means that musicexpr is transposed by the interval between the pitches frompitch and topitch: any note with pitch frompitch is changed to topitch and any other note is transposed by the same interval. Both pitches are entered in absolute mode.

Note: Music inside a \transpose block is absolute unless a \relative is included in the block.

Consider a piece written in the key of D-major. It can be transposed up to E-major; note that the key signature is automatically transposed as well.

\transpose d e {
  \relative c' {
    \key d \major
    d4 fis a d
  }
}

If a part written in C (normal concert pitch) is to be played on the A clarinet (for which an A is notated as a C and thus sounds a minor third lower than notated), the appropriate part will be produced with:

\transpose a c' {
  \relative c' {
    \key c \major
    c4 d e g
  }
}

Note that we specify \key c \major explicitly. If we do not specify a key signature, the notes will be transposed but no key signature will be printed.

\transpose distinguishes between enharmonic pitches: both \transpose c cis or \transpose c des will transpose up a semitone. The first version will print sharps and the notes will remain on the same scale step, the second version will print flats on the scale step above.
music = \relative c' \{ c d e f \}
\new Staff {
  \transpose c cis \{ \music \}
  \transpose c des \{ \music \}
}
\transpose may also be used in a different way, to input written notes for a transposing instrument. The previous examples show how to enter pitches in C (or concert pitch) and typeset them for a transposing instrument, but the opposite is also possible if you for example have a set of instrumental parts and want to print a conductor’s score. For example, when entering music for a B-flat trumpet that begins on a notated E (concert D), one would write:

\musicInBflat = \{ e4 \ldots \}
\transpose c bes, \musicInBflat

To print this music in F (e.g., rearranging to a French horn) you could wrap the existing music with another \transpose:

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

For more information about transposing instruments, see [Instrument transpositions], page 24.

Selected Snippets

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)))
        (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 (parser location 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 { }
}

See also

Notation Reference: [Instrument transpositions], page 24, [Inversion], page 13, [Modal transformations], page 14, [Relative octave entry], page 2, [Retrograde], page 13.

Snippets: Section “Pitches” in Snippets.

Known issues and warnings

The relative conversion will not affect \transpose, \chordmode or \relative sections in its argument. To use relative mode within transposed music, an additional \relative must be placed inside \transpose.

Triple accidentals will not be printed if using \transpose. An ‘enharmonically equivalent’ pitch will be used instead (e.g. d-flat rather than e-triple-flat).

Inversion

A music expression can be inverted and transposed in a single operation with:

\inversion around-pitch to-pitch musicexpr

The musicexpr is inverted interval-by-interval around around-pitch, and then transposed so that around-pitch is mapped to to-pitch.

\begin{verbatim}
\begin{music}
\new Staff {
  \inversion d' d' \music
  \inversion d' ees' \music
}\end{music}
\end{verbatim}

\begin{music}
\end{music}

Note: Motifs to be inverted should be expressed in absolute form or be first converted to absolute form by enclosing them in a \relative block.

See also

Notation Reference: [Modal transformations], page 14, [Retrograde], page 13, [Transpose], page 10.

Retrograde

A music expression can be reversed to produce its retrograde:

\begin{verbatim}
\begin{music}
\new Staff {
  \retrograde \music
}\end{music}
\end{verbatim}

\begin{music}
\end{music}

Known issues and warnings

Manual ties inside \retrograde will be broken and generate warnings. Some ties can be generated automatically by enabling [Automatic note splitting], page 72.
Modal transformations

In a musical composition that is based on a scale, a motif is frequently transformed in various ways. It may be transposed to start at different places in the scale or it may be inverted around a pivot point in the scale. It may also be reversed to produce its retrograde, see [Retrograde], page 13.

\textbf{Note:} Any note that does not lie within the given scale will be left untransformed.

\textbf{Modal transposition}

A motif can be transposed within a given scale with:

```
\modalTranspose from-pitch to-pitch scale motif
```

The notes of \texttt{motif} are shifted within the \texttt{scale} by the number of scale degrees given by the interval between \texttt{to-pitch} and \texttt{from-pitch}:

\begin{verbatim}
diatonicScale = \relative c' { c d e f g a b }
motif = \relative c' { c8 d e f g a b c }
\end{verbatim}

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

An ascending scale of any length and with any intervals may be specified:

\begin{verbatim}
pentatonicScale = \relative c' { ges aes bes des ees }
motif = \relative c' { ees8 des ges,4 <ges' bes,> <ges bes,> }
\end{verbatim}

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

When used with a chromatic scale \texttt{\modalTranspose} has a similar effect to \texttt{\transpose}, but with the ability to specify the names of the notes to be used:

\begin{verbatim}
chromaticScale = \relative c' { c cis d dis e f fis g gis a ais b }
motif = \relative c' { c8 d e f g a b c }
\end{verbatim}
Modal inversion
A motif can be inverted within a given scale around a given pivot note and transposed in a single operation with:
\modalInversion around-pitch to-pitch scale motif

The notes of motif are placed the same number of scale degrees from the around-pitch note within the scale, but in the opposite direction, and the result is then shifted within the scale by the number of scale degrees given by the interval between to-pitch and around-pitch.

So to simply invert around a note in the scale use the same value for around-pitch and to-pitch:

octatonicScale = \relative c' { ees f fis gis a b c d }
motif = \relative c' { c8. ees16 fis8. a16 b8. gis16 f8. d16 }

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

To invert around a pivot between two notes in the scale, invert around one of the notes and then transpose by one scale degree. The two notes specified can be interpreted as bracketing the pivot point:

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

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

The combined operation of inversion and retrograde produce the retrograde-inversion:

octatonicScale = \relative c' { ees f fis gis a b c d }
motif = \relative c' { c8. ees16 fis8. a16 b8. gis16 f8. d16 }

\new Staff {
  \motif
  \modalInversion c' g' \octatonicScale \motif
}


\begin{staff}
  \motif
  \retrograde \modalInversion c' c' \octatonicScale \motif
\end{staff}

See also
Notation Reference: [Inversion], page 13, [Retrograde], page 13, [Transpose], page 10.

1.1.3 Displaying pitches
This section discusses how to alter the output of pitches.

Clef
The clef may be altered. Middle C is shown in every example. The following clef names can (but do not need to) be enclosed in quotes.

\begin{verbatim}
c2 \c
\c treble
\c alto
\c tenor
\c bass
\c french
\c soprano
\c mezzosoprano
\c baritone
\c varbaritone
\c subbass
\c percussion
\end{verbatim}

Other clefs include:
\begin{verbatim}
c2 \c
\c french
\c soprano
\c mezzosoprano
\c baritone
\c varbaritone
\c subbass
\c percussion
\end{verbatim}
By adding \_8 or ^8 to the clef name, the clef is transposed one octave down or up respectively, and \_15 and ^15 transpose by two octaves. Other integers can be used if required. Clef names containing non-alphabetic characters must be enclosed in quotes.

```
\clef "treble_8"
c2 c
\clef "bass^15"
c2 c
\clef "alto_2"
c2 c
\clef "G_8"
c2 c
\clef "F^5"
c2 c
```

Optional octavation can be obtained by enclosing the numeric argument in parentheses or brackets:

```
\clef "treble_(8)"
c2 c
\clef "bass^[15]"
c2 c
```
The pitches are displayed as if the numeric argument were given without parentheses/brackets.

Some special purpose clefs are described in [Mensural clefs], page 409, [Gregorian clefs], page 417, [Default tablatures], page 318, and [Custom tablatures], page 331. For mixing different clefs when using cue notes within a score, see the \texttt{\textbackslash cueClef} and \texttt{\textbackslash cueDuringWithClef} functions in [Formatting cue notes], page 198.

**Selected Snippets**

**Tweaking clef properties**

The command \texttt{\textbackslash clef "treble\_8"} is equivalent to setting \texttt{clefGlyph}, \texttt{clefPosition} (which controls the vertical position of the clef), \texttt{middleCPosition} and \texttt{clefTransposition}. A clef is printed when any of the properties except \texttt{middleCPosition} are changed.

Note that changing the glyph, the position of the clef, or the octavation does not in itself change the position of subsequent notes on the staff: the position of middle C must also be specified to do this. In order to get key signatures on the correct staff lines, \texttt{middleCClefPosition} must also be set. The positional parameters are relative to the staff center line, positive numbers displacing upwards, counting one for each line and space. The \texttt{clefTransposition} value would normally be set to 7, -7, 15 or -15, but other values are valid.

When a clef change takes place at a line break the new clef symbol is printed at both the end of the previous line and the beginning of the new line by default. If the warning clef at the end of the previous line is not required it can be suppressed by setting the \texttt{Staff} property \texttt{explicitClefVisibility} to the value \texttt{end-of-line-invisible}. The default behavior can be recovered with \texttt{\textbackslash unset \textbackslash Staff.explicitClefVisibility}.

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.

\begin{verbatim}
\layout { ragged-right = ##t }
{
  % 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
\end{verbatim}
\set Staff.middleCPosition = #1
\set Staff.middleCClefPosition = #1
\key f \major
c'1
%
A non-standard clef
\set Staff.clefPosition = #0
\set Staff.clefTransposition = #0
\set Staff.middleCPosition = #-4
\set Staff.middleCClefPosition = #-4
\key g \major
c'1 \break
%
The following clef changes do not preserve
% the normal relationship between notes, key signatures
% and clefs:
\set Staff.clefGlyph = #"clefs.F"
\set Staff.clefPosition = #2
c'1
\set Staff.clefGlyph = #"clefs.G"
c'1
\set Staff.clefGlyph = #"clefs.C"
c'1
\set Staff.clefTransposition = #7
c'1
\set Staff.clefTransposition = #0
\set Staff.clefPosition = #0
c'1
%
Return to the normal clef:
\set Staff.middleCPosition = #0
c'1
}

See also
Notation Reference: [Mensural clefs], page 409, [Gregorian clefs], page 417, [Default tablatures], page 318, [Custom tablatures], page 331, [Formatting cue notes], page 198.

Snippets: Section “Pitches” in Snippets.

Internals Reference: Section “Clef_engraver” in Internals Reference, Section “Clef” in Internals Reference, Section “ClefModifier” in Internals Reference, Section “clef-interface” in Internals Reference.
Known issues and warnings

Ottavation numbers attached to clefs are treated as separate grobs. So any \override done to the Clef will also need to be applied, as a separate \override, to the ClefModifier grob.

Key signature

Note: New users are sometimes confused about accidentals and key signatures. In LilyPond, note names are the raw input; key signatures and clefs determine how this raw input is displayed. An unaltered note like c means ‘C natural’, regardless of the key signature or clef. For more information, see Section “Accidentals and key signatures” in Learning Manual.

The key signature indicates the tonality in which a piece is played. It is denoted by a set of alterations (flats or sharps) at the start of the staff. The key signature may be altered:

\key \pitch-mode

Here, \pitch-mode should be \major or \minor to get a key signature of \pitch-major or \pitch-minor, respectively. You may also use the standard mode names, also called church modes: \ionian, \dorian, \phrygian, \lydian, \mixolydian, \aeolian, and \locrian.

\key g \major fis1 f fis

Additional modes can be defined, by listing the alterations for each scale step when the mode starts on C.

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

\relative c' {
  \key c \freygish c4 des e f \\
  \bar "||" \key d \freygish d es fis g
}

Accidentals in the key signature may be printed in octaves other than their traditional positions, or in multiple octaves, by using the flat-positions and sharp-positions properties of KeySignature. Entries in these properties specify the range of staff-positions where accidentals will be printed. If a single position is specified in an entry, the accidentals are placed within the octave ending at that staff position.
\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 b2

\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 keySignature 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.keySignature = #`(((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.) The accidentals in the key signature will appear in the reverse order to that in which they are specified.

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 c' { 
\set Staff.keySignature = #`(((0 . 6) . ,FLAT) ((0 . 5) . ,FLAT) ((0 . 3) . ,SHARP)) 
c4 d e fis 
aes4 bes c2 
}

See also


Snippets: Section “Pitches” in Snippets.

Internals Reference: Section “KeyChangeEvent” in Internals Reference, Section “Key_engraver” in Internals Reference, Section “Key_performer” in Internals Reference, Section “KeyCancellation” in Internals Reference, Section “KeySignature” in Internals Reference, Section “key-signature-interface” in Internals Reference.

Ottava brackets

Ottava brackets introduce an extra transposition of an octave for the staff:

a2 b 
\ottava #-2 
a2 b 
\ottava #-1 
a2 b 
\ottava #0 
a2 b 
\ottava #1 
a2 b 
\ottava #2 
a2 b

Selected Snippets

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
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 ottavation 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.

\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

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

See also

Snippets: Section “Pitches” in Snippets.


### Instrument transpositions

When typesetting scores that involve transposing instruments, some parts can be typeset in a different pitch than the *concert pitch*. In these cases, the key of the *transposing instrument* should be specified; otherwise the MIDI output and cues in other parts will produce incorrect pitches. For more information about quotations, see [Quoting other voices], page 195.

\transposition \textit{pitch}

The pitch to use for \transposition should correspond to the real sound heard when a c' written on the staff is played by the transposing instrument. This pitch is entered in absolute mode, so an instrument that produces a real sound which is one tone higher than the printed music should use \transposition d'. \transposition should only be used if the pitches are not being entered in concert pitch.

Here are a few notes for violin and B-flat clarinet where the parts have been entered using the notes and key as they appear in each part of the conductor’s score. The two instruments are playing in unison.

\new GrandStaff <<
\new Staff = "violin" {
 \relative c'' {
  \set Staff.instrumentName = #'Vln
  \set Staff.midiInstrument = #'violin
  % not strictly necessary, but a good reminder
  \transposition c'
  \key c \major
  g4( c8) r c r c4
  }
}
\new Staff = "clarinet" {
 \relative c'' {
  \set Staff.instrumentName = \markup { Cl (B\flat) }
  \set Staff.midiInstrument = #'clarinet
  \transposition bes
  \key d \major
  a4( d8) r d r d4
  }
}
>>

![Musical notation example](image)

The \transposition may be changed during a piece. For example, a clarinetist may be required to switch from an A clarinet to a B-flat clarinet.

\textit{flute} = \relative c'' {
  \key f \major
  \cueDuring #'clarinet' #DOWN {

Chapter 1: Musical notation

See also


Notation Reference: [Quoting other voices], page 195, [Transpose], page 10.

Snippets: Section “Pitches” in Snippets.

Automatic accidentals

There are many different conventions on how to typeset accidentals. LilyPond provides a function to specify which accidental style to use. This function is called as follows:

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

The accidental style applies to the current Staff by default (with the exception of the styles piano and piano-cautionary, which are explained below). Optionally, the function can take a second argument that determines in which scope the style should be changed. For example, to use the same style in all staves of the current StaffGroup, use:

   \accidentalStyle StaffGroup.voice

The following accidental styles are supported. To demonstrate each style, we use the following example:
musicA = {
  <<
  \relative c' {
      cis'8 fis, bes4 <a cis>8 f bis4 |
      cis2. <c, g'>4 |
  }
  \\
  \relative c' {
      ais'2 cis, |
      fis8 b a4 cis2 |
  }
  >>
}

musicB = {
  \clef bass
  \new Voice {
    \voiceTwo \relative c' {
      <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 
  }
  >>
}
Note that the last lines of this example can be replaced by the following, as long as the same accidental style should be used in both staves.

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

This is the default typesetting behavior. It corresponds to eighteenth-century common practice: accidentals are remembered to the end of the measure in which they occur and only in their own octave. Thus, in the example below, no natural signs are printed before the b in the second measure or the last c:

![Example notation](image1.png)

**voice**

The normal behavior is to remember the accidentals at Staff-level. In this style, however, accidentals are typeset individually for each voice. Apart from that, the rule is similar to default.

As a result, accidentals from one voice do not get canceled in other voices, which is often an unwanted result: in the following example, it is hard to determine whether the second a should be played natural or sharp. The voice option should therefore be used only if the voices are to be read solely by individual musicians. If the staff is to be used by one musician (e.g., a conductor or in a piano score) then modern or modern-cautionary should be used instead.

![Example notation](image2.png)

**modern**

This rule corresponds to the common practice in the twentieth century. It omits some extra natural signs, which were traditionally prefixed to a sharp following a double sharp, or a flat following a double flat. The modern rule prints the same accidentals as default, with two additions that serve to avoid ambiguity: after
temporary accidentals, cancellation marks are printed also in the following measure
(for notes in the same octave) and, in the same measure, for notes in other octaves.
Hence the naturals before the b and the c in the second measure of the upper staff:

![Musical notation example](image)

**modern-cautionary**

This rule is similar to **modern**, but the ‘extra’ accidentals (the ones not typeset
by **default**) are typeset as cautionary accidentals. They are by default printed
with parentheses, but they can also be printed in reduced size by defining the
cautionary-style property of **AccidentalSuggestion**.

![Musical notation example](image)

**modern-voice**

This rule is used for multivoice accidentals to be read both by musicians playing
one voice and musicians playing all voices. Accidentals are typeset for each voice,
but they are canceled across voices in the same **Staff**. Hence, the a in the last
measure is canceled because the previous cancellation was in a different voice, and
the d in the lower staff is canceled because of the accidental in a different voice in
the previous measure:

![Musical notation example](image)

**modern-voice-cautionary**

This rule is the same as **modern-voice**, but with the extra accidentals (the ones not
typeset by **voice**) typeset as cautionaries. Even though all accidentals typeset by
**default** are typeset with this rule, some of them are typeset as cautionaries.

![Musical notation example](image)

**piano**
This rule reflects twentieth-century practice for piano notation. Its behavior is very similar to modern style, but here accidentals also get canceled across the staves in the same GrandStaff or PianoStaff, hence all the cancellations of the final notes. This accidental style applies to the current GrandStaff or PianoStaff by default.

**piano-cautionary**

This is the same as piano but with the extra accidentals typeset as cautionaries.

**neo-modern**

This rule reproduces a common practice in contemporary music: accidentals are printed like with modern, but they are printed again if the same note appears later in the same measure – except if the note is immediately repeated.

**neo-modern-cautionary**

This rule is similar to neo-modern, but the extra accidentals are printed as cautionary accidentals.

**neo-modern-voice**

This rule is used for multivoice accidentals to be read both by musicians playing one voice and musicians playing all voices. Accidentals are typeset for each voice as with neo-modern, but they are canceled across voices in the same Staff.
**neo-modern-voice-cautionary**

This rule is similar to neo-modern-voice, but the extra accidentals are printed as cautionary accidentals.

**dodecaphonic**

This rule reflects a practice introduced by composers at the beginning of the 20th century, in an attempt to abolish the hierarchy between natural and non-natural notes. With this style, *every* note gets an accidental sign, including natural signs.

**teaching**

This rule is intended for students, and makes it easy to create scale sheets with automatically created cautionary accidentals. Accidentals are printed like with modern, but cautionary accidentals are added for all sharp or flat tones specified by the key signature, except if the note is immediately repeated.

**no-reset**

This is the same as default but with accidentals lasting ‘forever’ and not only within the same measure:

**forget**
This is the opposite of no-reset: Accidentals are not remembered at all – and hence all accidentals are typeset relative to the key signature, regardless of what came before in the music.

See also

Snippets: Section “Pitches” in Snippets.

Internals Reference: Section “Accidental” in Internals Reference, Section “Accidental_engraver” in Internals Reference, Section “GrandStaff” in Internals Reference, Section “PianoStaff” in Internals Reference, Section “Staff” in Internals Reference, Section “AccidentalSuggestion” in Internals Reference, Section “AccidentalPlacement” in Internals Reference, Section “piano staff” in Internals Reference, Section “staves” in Internals Reference, Section “accidental-suggestion-interface” in Internals Reference.

Known issues and warnings

Simultaneous notes are not considered in the automatic determination of accidentals; only previous notes and the key signature are considered. Forcing accidentals with ! or ? may be required when the same note name occurs simultaneously with different alterations, as in ‘<f! fis!>’.

Cautionary cancellation of accidentals is done by looking at previous measure. However, in the \alternative block following a \repeat volta \# section, one would expect the cancellation being calculated using the previous played measure, not previous printed measure. In the following example, the natural c in the second alternative does not need a natural sign:

The following work-around can be used: define a function that locally changes the accidental style to forget:

```lilypond
forget = #(define-music-function (parser location music) (ly:music?) {\
  \accidentalStyle forget
  #music
  \accidentalStyle modern
#})
{
  \accidentalStyle modern
  \time 2/4
  \repeat volta 2 {c'2}
}\alternative {
  cis'
  \forget c'
}
```
Chapter 1: Musical notation

Ambitus

The term *ambitus* (pl. ambitus) denotes a range of pitches for a given voice in a part of music. It may also denote the pitch range that a musical instrument is capable of playing. Ambitus are printed on vocal parts so that performers can easily determine if it matches their capabilities.

Ambitus are denoted at the beginning of a piece near the initial clef. The range is graphically specified by two note heads that represent the lowest and highest pitches. Accidentals are only printed if they are not part of the key signature.

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

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

Selected Snippets

*Adding ambitus per voice*

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

```latex
\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''
}

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

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

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

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

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

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

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

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

See also


Snippets: Section “Pitches” in Snippets.


Known issues and warnings

There is no collision handling in the case of multiple per-voice ambitus.

1.1.4 Note heads

This section suggests ways of altering note heads.

Special note heads

The appearance of note heads may be altered:

\begin{verbatim}
c4 b
\override NoteHead.style = #'cross
c4 b
\end{verbatim}

\begin{verbatim}
a b
\override NoteHead.style = #'harmonic
a b
\end{verbatim}

\begin{verbatim}
c4 d e f
\end{verbatim}
To see all note head styles, see Section A.9 [Note head styles], page 645.

The \textit{cross} style is used to represent a variety of musical intentions. The following generic predefined commands modify the note head in both staff and tablature contexts and can be used to represent any musical meaning:

\begin{verbatim}
c4 b \xNotesOn a b c4 b \xNotesOff c4 d
\end{verbatim}

The music function form of this predefined command may be used inside and outside chords to generate crossed note heads in both staff and tablature contexts:

\begin{verbatim}
c4 b \xNote { e f } c b < g \xNote c f > b
\end{verbatim}

As synonyms for $\xNote$, $\xNotesOn$ and $\xNotesOff$, $\texttt{deadNote}$, $\texttt{deadNotesOn}$ and $\texttt{deadNotesOff}$ can be used. The term \textit{dead note} is commonly used by guitarists.

There is also a similar shorthand for diamond shapes:

\begin{verbatim}
< c \harmonic> 2 < d a' \harmonic> 4 < c g' \harmonic> f \harmonic
\end{verbatim}

\section*{Predefined commands}

$\texttt{\harmonic}$, $\texttt{\NotesOn}$, $\texttt{\NotesOff}$, $\texttt{\Note}$.

\section*{See also}

\textit{Snippets}: Section “Pitches” in \textit{Snippets}.

\textit{Notation Reference}: Section A.9 [Note head styles], page 645, [Chorded notes], page 154, [Indicating harmonics and dampened notes], page 358.

Easy notation note heads

The ‘easy play’ note head includes a note name inside the head. It is used in music for beginners. To make the letters readable, it should be printed in a large font size. To print with a larger font, see Section 4.2.2 [Setting the staff size], page 506.

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

Predefined commands
\( \texttt{\easyHeadsOn}, \texttt{\easyHeadsOff}. \)

Selected Snippets

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.

\( \texttt{#(define Ez_numbers_engraver} \)
\( \texttt{(make-engraver} \)
\( \texttt{(acknowledgers} \)
\( \texttt{((note-head-interface engraver grob source-engraver))} \)
\( \texttt{(let* ((context (ly:translator-context engraver))} \)
\( \texttt{(tonic-pitch (ly:context-property context 'tonic))} \)
\( \texttt{(tonic-name (ly:pitch-notename tonic-pitch))} \)
\( \texttt{(grob-pitch} \)
\( \texttt{(ly:event-property (event-cause grob) 'pitch))} \)
\( \texttt{(grob-name (ly:pitch-notename grob-pitch))} \)
\( \texttt{(delta (modulo (- grob-name tonic-name) 7))} \)
\( \texttt{(note-names} \)
\( \texttt{(make-vector 7 (number->string (1+ delta))))}) \)
\( \texttt{(ly:grob-set-property! grob 'note-names note-names)))}} \)
\( \}\)
\( \texttt{#(set-global-staff-size 26)} \)
\( \texttt{\layout} { } \)
\( \texttt{ragged-right = ##t} \)
\( \texttt{\context} { } \)
\( \texttt{\Voice} \)
\( \texttt{\consists \Ez_numbers_engraver} \)
\( \}\)
Shape note heads

In shape note head notation, the shape of the note head corresponds to the harmonic function of a note in the scale. This notation was popular in nineteenth-century American song books. Shape note heads can be produced in Sacred Harp, Southern Harmony, Funk (Harmonica Sacra), Walker, and Aiken (Christian Harmony) styles:

\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

See also

Notation Reference: Section 4.2.2 [Setting the staff size], page 506.
Snippets: Section “Pitches” in Snippets.
Internals Reference: Section “note-event” in Internals Reference, Section “Note_heads_engraver” in Internals Reference, Section “NoteHead” in Internals Reference, Section “note-head-interface” in Internals Reference.
Shapes are typeset according to the step in the scale, where the base of the scale is determined by the \key command. When writing in a minor key, the scale step can be determined from the relative major:

\key a \minor
\aikenHeads
a,4 d e f g2 a b1 c \break
\aikenHeadsMinor
a,4 d e f g2 a b1 c \break
\sacredHarmonyHeadsMinor
a,2 b c d \break
\southernHarmonyHeadsMinor
a2 b c d \break
\funkHeadsMinor
a2 b c d \break
\walkerHeadsMinor
a2 b c d \break
Selected Snippets

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.

```plaintext
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
}
\fragment

To see all note head styles, see Section A.9 [Note head styles], page 645.

See also

Snippets: Section “Pitches” in Snippets.
Notation Reference: Section A.9 [Note head styles], page 645.
Internals Reference: Section “note-event” in Internals Reference, Section
“Note_heads_engraver” in Internals Reference, Section “NoteHead” in Internals
Reference, Section “note-head-interface” in Internals Reference.

Improvisation

Improvisation is sometimes denoted with slashed note heads, where the performer may choose
any pitch but should play the specified rhythm. Such note heads can be created:
\new Voice \with {
  \consists "Pitch_squash_engraver"
} {
  e8 e g a a16( bes) a8 g
  \improvisationOn
  e8 ~
  e2 ~ e8 f4 f8 ~
  f2
  \improvisationOff
  a16( bes) a8 g e
}

Predefined commands
\improvisationOn, \improvisationOff.

See also

Snippets: Section “Pitches” in Snippets.
Internals Reference: Section “Pitch_squash_engraver” in Internals Reference, Section “Voice”
in Internals Reference, Section “RhythmicStaff” in Internals Reference.
1.2 Rhythms

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.

\time 8/1
\c\longa c \breve c1 c2
c4 c8 c16 c32 c64 c128 c128
Here are the same durations with automatic beaming turned off.
\begin{verbatim}
\time 8/1
\autoBeamOff
c\longa c\breve c1 c2
c4 c8 c16 c32 c64 c128 c128
\end{verbatim}

A note with the duration of a quadruple breve may be entered with \texttt{\maxima}, but this is supported only within ancient music notation. For details, see Section 2.9 [Ancient notation], page 404.

If the duration is omitted, it is set to the previously entered duration. The default for the first note is a quarter note.
\begin{verbatim}
a a a2 a a4 a a1 a
\end{verbatim}

To obtain dotted note lengths, place a dot (.) after the duration. Double-dotted notes are specified by appending two dots, and so on.
\begin{verbatim}
a4 b c4. b8 a4. b4.. c8.
\end{verbatim}

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], page 49.

For ways of specifying durations for the syllables of lyrics and ways of aligning lyrics to notes, see Section 2.1 [Vocal music], page 239.

Optionally, notes can be spaced strictly proportionately to their duration. For details of this and other settings which control proportional notation, see Section 4.5.5 [Proportional notation], page 535.

Dots are normally moved up to avoid staff lines, except in polyphonic situations. Dots may be manually placed above or below the staff; see Section 5.4.2 [Direction and placement], page 577.

**Predefined commands**
\texttt{\autoBeamOn}, \texttt{\autoBeamOff}, \texttt{\dotsUp}, \texttt{\dotsDown}, \texttt{\dotsNeutral}.

**Selected Snippets**

*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.
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' {
\relative c'' {
\time 4/2
\override Staff.NoteHead.style = \"altdefault\" 
\breve b \breve 
\override Staff.NoteHead.style = \"baroque\" 
\breve b \breve 
\revert Staff.NoteHead.style
\breve a
}
\override Dots.dot-count = #4
\override Dots.dot-count = #0
\override Dots.dot-count
\override Dots.dot-count
\override Dots.dot-count
\revert Dots.dot-count
\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
\relative c' {
c4.. a16 r2 |
}
```

See also


Notation Reference: [Automatic beams], page 76, [Ties], page 49, [Stems], page 210, Section 1.2.1 [Writing rhythms], page 41, Section 1.2.2 [Writing rests], page 52, Section 2.1 [Vocal music], page 239, Section 2.9 [Ancient notation], page 404, Section 4.5.5 [Proportional notation], page 535.

Snippets: Section “Rhythms” in Snippets.

Internals Reference: Section “Dots” in Internals Reference, Section “DotColumn” in Internals Reference.

Known issues and warnings

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 x whole) may be printed.
Tuplets

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

\texttt{tuplet \textit{fraction} \{ \textit{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:

\begin{verbatim}
a2 \texttt{tuplet} 3/2 \{ b4 b b \}
c4 \texttt{c} \texttt{tuplet} 3/2 \{ b4 a g \}
\end{verbatim}

When entering long passages of tuplets, having to write a separate \texttt{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:

\begin{verbatim}
g2 r8 \texttt{tuplet} 3/2 8 \{ cis16 d e e f g g f e \}
\end{verbatim}

Tuplet brackets may be manually placed above or below the staff; see Section 5.4.2 [Direction and placement], page 577.

Tuplets may be nested:

\begin{verbatim}
\texttt{autoBeamOff}
c4 \texttt{tuplet} 5/4 \{ f8 e f \texttt{tuplet} 3/2 \{ e[ f g] \} \} \texttt{f4}
\end{verbatim}

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

To modify the duration of notes without printing a tuplet bracket, see [Scaling durations], page 48.

Predefined commands
\texttt{tupletUp}, \texttt{tupletDown}, \texttt{tupletNeutral}.

Selected Snippets

\textit{Entering several tuplets using only one \texttt{tuplet} command}

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

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

By default, only the numerator of the tuplet number is printed over the tuplet bracket. Alternatively, num:den of the tuplet number may be printed, or the tuplet number may be suppressed altogether.

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.
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).

```latex
\begin{verbatim}
music = \relative c'' {
  \tuplet 3/2 { c16 d e f8 }
  \tuplet 3/2 { c8 d e }
  \tuplet 3/2 { c4 d e }
}
\end{verbatim}
```

```latex
\begin{verbatim}
\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" >>
  }
}
```

```latex
\begin{verbatim}
...}
```

```latex
\begin{verbatim}
...}
```

```latex
\begin{verbatim}
...}
```

```latex
\begin{verbatim}
...}
```

```latex
\begin{verbatim}
...}
```

```latex
\begin{verbatim}
...}
```
Permitting line breaks within beamed tuplets

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

\layout {
  \context {
    \Voice
    % Permit line breaks within tuplets
    \remove "Forbid_line_break_engraver"
    % Allow beams to be broken at line breaks
    \override Beam.breakable = ##t
  }
}
\relative c'' {
  a8
  \repeat unfold 5 { \tuplet 3/2 { c[ b a ] } }
  % Insert a manual line break within a tuplet
  \tuplet 3/2 { c[ b \bar "" \break a ] }
  \repeat unfold 5 { \tuplet 3/2 { c[ b a ] } }
  c8
}

See also


Notation Reference: [Time administration], page 110, [Scaling durations], page 48, Section 5.3.4 [The tweak command], page 571, [Polymetric notation], page 69.

Snippets: Section “Rhythms” in Snippets.

Known issues and warnings

Grace notes may be placed within tuplet brackets, *except* when a staff begins with a grace note followed by a tuplet. In this particular case, the grace note must be placed before the `\tuplet` command to avoid errors.

When using a tuplet at the beginning of a piece with a `\tempo` mark, the music must be explicitly entered in a `\new Voice` block, as discussed in Section “Voices contain music” in Learning Manual.

Scaling durations

The duration of single notes, rests or chords may be multiplied by a fraction \( \frac{N}{M} \) 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.

\[
\text{\time 2/4} \\
% Alter durations to triplets \\
a4*2/3 gis a \\
% Normal durations \\
a4 a \\
% Double the duration of chord \\
<a d>4\times2 \\
% Duration of quarter, appears like sixteenth \\
b16*4 c4
\]

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:

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

One application of this command is in polymetric notation, see [Polymetric notation], page 69.

**See also**

Notation Reference: [Tuplets], page 44, [Invisible rests], page 54, [Polymetric notation], page 69.

Snippets: Section “Rhythms” in Snippets.

**Known issues and warnings**

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.

**Note:** 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.

```
\relative c' {
  r8 c~ c2 r4 |
  r8 "not" c2~ c8 r4
}
```

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 c' {
  r8 c~ c2 r4 |
  r8 "not" c2~ c8 r4
}
```

If you need to tie many notes across bar lines, it may be easier to use automatic note splitting, see [Automatic note splitting], page 72. 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.
When a second alternative of a repeat starts with a tied note, you have to specify the repeated tie as follows:

```
\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 Section 5.4.2 [Direction and placement], page 577.

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

```
\tieDotted
c2~ c
\tieDashed
c2~ c
\tieHalfDashed
c2~ c
\tieHalfSolid
c2~ c
\tieSolid
c2~ c
```

Custom dash patterns can be specified:
Dash pattern definitions for ties have the same structure as dash pattern definitions for slurs. For more information about complex dash patterns, see [Slurs], page 122.

Override whiteout and layer layout properties for ties that collide with other objects in a staff.

\override Tie.layer = #-2
\override Staff.TimeSignature.layer = #-1
\override Staff.KeySignature.layer = #-1
\override Staff.TimeSignature.whiteout = ##t
\override Staff.KeySignature.whiteout = ##t
b2 b~
\time 3/4
\key a \major
b r4

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

Selected Snippets

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

See also

Notation Reference: [Slurs], page 122, [Automatic note splitting], page 72.
Snippets: Section “Expressive marks” in Snippets, Section “Rhythms” in Snippets.

Known issues and warnings

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], page 55.

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.

```
a4\rest d4\rest
```

Selected Snippets

*Rest styles*

Rests may be used in various styles.

```
\layout {
  indent = 0
  \context {
    \Staff
    \remove "Time_signature_engraver"
  }
}

\new Staff \relative c {
  \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 ""

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

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

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

\mensural
```
See also


Notation Reference: [Full measure rests], page 55.
Snippets: Section “Rhythms” in Snippets.
Internals Reference: Section “Rest” in Internals Reference.

Known issues and warnings

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 x whole).

Invisible rests

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

\[ \text{c4 c s c} \]
\[ \text{s2 c} \]

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

\begin{verbatim}
<<
  a2 \texttt{\textbackslash skip}2 a2 a2
}\new Lyrics {
  \texttt{\textbackslash lyricmode} {
    foo2 \texttt{\textbackslash skip} 1 bla2
  }
}
>>
\end{verbatim}

Because \texttt{\textbackslash skip} is a command, it does not affect the default durations of following notes, unlike s.
A spacer rest implicitly causes Staff and Voice contexts to be created if none exist, just like notes and rests do:

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

% This is valid input, but does nothing
\skip 1 \skip 1 \skip 1

See also

Notation Reference: [Hidden notes], page 207, Section 5.4.6 [Visibility of objects], page 584.
Snippets: Section “Rhythms” in Snippets.
Internals Reference: Section “SkipMusic” in Internals Reference.

Full measure rests

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

% Rest measures contracted to single measure
\compressFullBarRests
R1\*4
R1\*24
R1\*4
b2"Tutti" b4 a4

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

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:

Markups can be added to multi-measure rests. The predefined command \fermataMarkup is provided for adding fermatas.
Chapter 1: Musical notation

R2."\fermataMarkup

ad lib.

\clefs.G

\timesig.C44

\rests.0

\rests.M3

\rests.M1

\textLengthOn, \textLengthOff, \fermataMarkup, \compressFullBarRests, \expandFullBarRests.

Selected Snippets

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'' { %
\compressFullBarRests
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' {  
    \override MultiMeasureRest.staff-position = #-2  
    \override MultiMeasureRest.staff-position = #0  
    \override MultiMeasureRest.staff-position = #2  
    \override MultiMeasureRest.staff-position = #3  
    \override MultiMeasureRest.staff-position = #6  
    \revert MultiMeasureRest.staff-position  
    \break  
    \override MultiMeasureRest.staff-position = #0  
    \compressFullBarRests  
    \revert MultiMeasureRest.direction  
    { R1*3 }  
    \revert MultiMeasureRest.direction  
    { R1*3 }  
}\relative c' {  
    \override MultiMeasureRest.staff-position = #-2  
    \override MultiMeasureRest.staff-position = #0  
    \override MultiMeasureRest.staff-position = #2  
    \override MultiMeasureRest.staff-position = #3  
    \override MultiMeasureRest.staff-position = #6  
    \revert MultiMeasureRest.staff-position  
    \break  
    \override MultiMeasureRest.staff-position = #0  
    \compressFullBarRests  
    \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' {
  \compressFullBarRests
  \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
}

Cue: ... it is yours

See also

Notation Reference: [Durations], page 41, Section 1.8 [Text], page 215, Section 1.8.2 [Formatting text], page 223, [Text scripts], page 216.
Snippets: Section “Rhythms” in *Snippets*.
Internals Reference: Section “MultiMeasureRest” in *Internals Reference*, Section “MultiMeasureRestNumber” in *Internals Reference*, Section “MultiMeasureRestText” in *Internals Reference*.

Known issues and warnings

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 c2
\time 3/4 c2.
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 Section 5.4.6 [Visibility of objects], page 584.

\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:

% 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], page 410.

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:
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], page 78.

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

```
\score{
  \relative c' {
    \repeat unfold 8 { c8 } |
  }
}
```
Different values of default time signature properties can be established for different staves by moving the TimingTranslator and the Default_bar_line_engraver from the Score context to the Staff context.
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], page 78.

Predefined commands
\numericTimeSignature, \defaultTimeSignature.

Selected Snippets

**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  
}
See also

Music Glossary: Section “time signature” in Music Glossary
Notation Reference: [Mensural time signatures], page 410, [Setting automatic beam behavior], page 78, [Time administration], page 110.
Installed Files: ‘scm/time-signature-settings.scm’.
Snippets: Section “Rhythms” in Snippets.

Metronome marks

A basic metronome mark is simple to write:

\tempo 4 = 120
c2 d
e4. d8 c2

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

\tempo 4 = 40 - 46
c4. e8 a4 g
b,2 d4 r

Tempo indications with text can be used instead:

\tempo "Allegretto"
c4 e d c
b4. a16 b c4 r4

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

\tempo "Allegro" 4 = 160
g4 c d e
d4 b g2

In general, the text can be any markup object:
A parenthesized metronome mark with no textual indication may be written by including an empty string in the input:
\tempo "" 8 = 96
d4 g e c
\( \dot{\downarrow} = 96 \)
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
    }
  }
  \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 Section 1.8.2 [Formatting text], page 223.

See also

Notation Reference: Section 1.8.2 [Formatting text], page 223, Section 3.5 [MIDI output], page 483.

Snippets: Section “Staff notation” in Snippets.

Internals Reference: Section “MetronomeMark” in Internals Reference.

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

\partial duration

where duration is the remaining length of the partial measure before the start of the next full measure.

\time 3/4
\partial 8
\start staff
  e8 | a4 c8 b c4 |
\stop staff

The duration can be any value less than a full measure:

\time 3/4
\partial 4.
\start staff
  r4 e8 | a4 c8 b c4 |
\stop staff

\partial duration can also be written as:

\set Timing.measurePosition -duration

So the first example above could be written:

\time 3/4
\set Timing.measurePosition = #(ly:make-moment -1/8)
\start staff
  e8 | a4 c8 b c4 |
\stop staff

The property measurePosition contains a rational number, which is usually positive and indicates how much of the measure has passed at this point. The \partial duration command sets it to a negative number, when it has a different meaning: it then says that the current (first) bar will be preceded by a bar 0 (the partial bar) with a duration given by duration.
See also

Notation Reference: [Grace notes], page 104.
Snippets: Section “Rhythms” in Snippets.
Internal Reference: Section “Timing translator” in Internals Reference.

Known issues and warnings

The \partial command should be used only at the beginning of a piece. If you use it after the beginning, warnings or problems may occur, so use \set Timing.measurePosition instead.

\time 6/8
\partial 8
 e8 | a4 c8 b[ c b] |
\set Timing.measurePosition = #(ly:make-moment -1/4)
 r8 e,8 | a4 c8 b[ c b] |

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.

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.

% 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], page 5.

\begin{verbatim}
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 "|
\end{verbatim}

Automatic beaming is disabled by \cadenzaOn. Therefore, all beaming in cadenzas must be entered manually. See [Manual beams], page 86.

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

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], page 69.

**Predefined commands**

\cadenzaOn, \cadenzaOff.

**See also**


Notation Reference: Section 5.4.6 [Visibility of objects], page 584, [Polymetric notation], page 69, [Manual beams], page 86, [Accidentals], page 5.

Snippets: Section “Rhythms” in Snippets.

**Known issues and warnings**

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 unmetered music to permit breaking:

\begin{verbatim}
\bar ""
\end{verbatim}

**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 c' <<
\new Staff {
  \time 3/4
c4 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"
\textbf{Compound time signatures}

These are created using the \texttt{\compoundMeter} function. The syntax for this is:

\begin{verbatim}
\compoundMeter #'(list of lists)
\end{verbatim}

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.

\begin{verbatim}
\relative c' {
  \compoundMeter #'((2 2 2 8))
  \repeat unfold 6 c8 \repeat unfold 12 c16
}
\end{verbatim}
More complex meters can be constructed using additional lists. Also, automatic beaming settings will be adjusted depending on the values.

\relative c' {
    \compoundMeter #'((1 4) (3 8))
    \repeat unfold 5 c8 \repeat unfold 10 c16
}

\relative c' {
    \compoundMeter #'((1 2 3 8) (3 4))
    \repeat unfold 12 c8
}

See also


Notation Reference: [Automatic beams], page 76, [Manual beams], page 86, [Time signature], page 59, [Scaling durations], page 48.

Snippets: Section “Rhythms” in Snippets.

Internals Reference: Section “TimeSignature” in Internals Reference, Section “Timing_translator” in Internals Reference, Section “Default_bar_line_engraver” in Internals Reference, Section “Staff” in Internals Reference.

Known issues and warnings

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"
}
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.

See also

Music Glossary: Section “tie” in Music Glossary
Snippets: Section “Rhythms” in Snippets.
Internals Reference: Section “Note_heads_engraver” in Internals Reference, Section “Completion_heads_engraver” in Internals Reference, Section “Rest_engraver” in Internals Reference, Section “Completion_rest_engraver” in Internals Reference, Section “Forbid_line_break_engraver” in Internals Reference.

Known issues and warnings

Not all durations (especially those containing tuplets) can be represented exactly with normal notes and dots, but the Completion_heads_engraver will not insert tuplets.

The Completion_heads_engraver only affects notes; it does not split rests.

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" {
    \time 4/4
    c4 e8 f g2
    r4 g g f
    g1
  }
}
\new Lyrics {
  \lyricsto "myRhythm" {
    This is my song
    I like to sing
  }
}

\score { c2. c8 d4 e f g a b c8 c2 b4 a g16 f4 e d c8. c2 r1*2 }
Chapter 1: Musical notation

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
} \improvisationOff

Predefined commands
\improvisationOn, \improvisationOff.

Selected Snippets

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
  }
}
See also

Snippets: Section “Rhythms” in Snippets.

Internals Reference: Section “RhythmicStaff” in Internals Reference, Section “Pitch_squash_engraver” in Internals Reference.

1.2.4 Beams
Automatic beams

By default, beams are inserted automatically:

\begin{verbatim}
\time 2/4 c8 c c c
\time 6/8 c8 c c c8. c16 c8
\end{verbatim}

If these automatic decisions are not satisfactory, beaming can be entered explicitly; see [Manual beams], page 86. Beams 	extit{must} be entered manually if beams are to be extended over rests.

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

\begin{verbatim}
c4 c8 c8. c16 c8. c16 c8
\autoBeamOff
c4 c8 c8. c16 c8.
\autoBeamOn
c16 c8
\end{verbatim}

\textbf{Note:} If beams are used to indicate melismata in songs, then automatic beaming should be switched off with \texttt{\autoBeamOff} and the beams indicated manually. Using \texttt{\partcombine} with \texttt{\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], page 78.

Predefined commands
\texttt{\autoBeamOff, \autoBeamOn}.

Selected Snippets

\textit{Beams across line breaks}

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

\begin{verbatim}
\relative c'' {
  \override Beam.breakable = ##t
  c8 c[ c] c[ c] c[ c] c[ \break
  c8] c[ c] c[ c] c[ c] c
}
\end{verbatim}
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 
adjusted 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 |
  }
}%
```
See also

Notation Reference: [Manual beams], page 86, [Setting automatic beam behavior], page 78.
Installed Files: ‘scm/auto-beam.scm’.
Snippets: Section “Rhythms” in Snippets.

Known issues and warnings

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], page 59.

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.

\time 5/16
\set Timing.beamExceptions = #'()
\set Timing.beatStructure = #'(2 3)
c16"'(2+3)" c c c c |
c16"'(3+2)" c c c c |
\time 5/16
\set Timing.beamExceptions = #'()
\set Timing.beatStructure = #'(2 3)
c16"'(2+3)" c c c c |
c16"'(3+2)" c c c c |
\time 5/16
\set Timing.beamExceptions = #'()
\set Timing.beatStructure = #'(2 3)
c16"'(2+3)" c c c c |
c16"'(3+2)" c c c c |
\time 4/4
a8"default" a 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 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 c'' {
  a8 a a a a a a
}
}
\new Voice = two {
\relative c' {
  \voiceTwo
  \set Voice.beatStructure = #'(1 3 3)
  f8 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)
<< \{a8 a a a16 a a a8 a\} \\ \{f4. f8 f f f\} >>

\% Works correctly with context Staff specified
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.

```
\set Staff.beatStructure = #'(3 1 1 2)
<< {a8 a a a16 a a a a8 a} \ {f4. f8 f f f} >>
```


\set Timing.baseMoment = #(ly:make-moment 1/16)
\set Timing.beatStructure = #'(7 3)
\repeat unfold 10 { a16 }

**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], page 110.

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

```
\set Timing.beatStructure = #'(7 3)
\set Timing.beamExceptions =
  #'(end .
    (end .
      (1 . 32) . (2 2 2)))
\repeat unfold 6 { c32 } |
```

beamExceptions is an alist with a key of rule-type and a value of beaming-rules. At this time the only available value of rule-type is *end* for beam ending.

Beaming-rules is a scheme alist (or list of pairs) that indicates the beam type and the grouping to be applied to beams containing notes with a shortest duration of that beam type.
Beam type is a scheme pair indicating the duration of the beam, e.g., (1 . 16).

Grouping is a scheme list indicating the grouping to be applied to the beam. The grouping is in units of the beam type.

Note: 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.

\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 }
\begin{music}
    % group (4 + 2)
\end{music}

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], page 59.

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 {c8} |
% This will beam (3 3 2) because we clear beamExceptions
\set Timing.beamExceptions = #'()
\repeat unfold 8 {c8}
\begin{music}
    % group (4 + 2)
\end{music}

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.
% by default we beam in (6) due to beamExceptions
\repeat unfold 6 {a8} |
% This will beam (1 1 1) due to default baseMoment and beatStructure
\set Timing.beamExceptions = #'()
\repeat unfold 6 {a8}

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:
\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’.

Selected Snippets

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 just one beam between the sub-groups. 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:
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
```
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
}
\new Staff {
  % Modify beaming for just this staff
  \set Staff.beatStructure = #'(6 4)
  c8 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 a
}

\new Staff {
\time 3/4
\set Timing.baseMoment = #(ly:make-moment 1/8)
\set Timing.beatStructure = #'(1 5)
\repeat unfold 6 { a8 }
}
\new Staff {
\repeat unfold 6 { a8 }
}

See also
Notation Reference: [Time signature], page 59.
Installed Files: 'scm/time-signature-settings.scm'.
Snippets: Section “Rhythms” in Snippets.

Known issues and warnings
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)
\repeat unfold 6 { a8 }
}
\new Staff {
\repeat unfold 6 { a8 }
}

>>
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], page 59.

```
<<
\new Staff {
    \overrideTimeSignatureSettings
    3/4 % timeSignatureFraction
    1/8 % baseMomentFraction
    #'(1 5) % beatStructure
    #'(0) % beamExceptions

\time 3/4
\repeat unfold 6 { a8 }
}

\new Staff {
    \time 3/4
    \repeat unfold 6 { a8 }
}
>>
```

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 ].

```
r4 r8[ g' a r] r g[ | a] r
```

Beaming direction can be set manually using direction indicators:

```
c8^-[ d e] c,-[ d e f g]
```

Individual notes may be marked with \noBeam to prevent them from being beamed:
Grace note beams and normal note beams can occur simultaneously. Unbeamed grace notes are not put into normal note beams.

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.

Predefined commands

\noBeam.

Selected Snippets

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 \[\text{beam indicators alone (Example 4).}\]

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

\begin{verbatim}
\score {
  \new RhythmicStaff {
    \set stemLeftBeamCount = #0
c16[]
r8.
  }
}
\begin{verbatim}
% Example 1
\new RhythmicStaff {
  \set stemLeftBeamCount = #0
c16[]
r8.
}
% Example 2
\new RhythmicStaff {
  r8.
  \set stemRightBeamCount = #0
c16[]
}
% Example 3
\new RhythmicStaff {
  c16 c
  \set stemRightBeamCount = #2
c16 r r
  \set stemLeftBeamCount = #2
c16 c c
}
% Example 4
\new RhythmicStaff {
  c16 c
  \set stemRightBeamCount = #2
c16 r
c16[]
r16
  \set stemLeftBeamCount = #2
c16 c
}
  >>
}
\end{verbatim}
Chapter 1: Musical notation

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.grow-direction` property.

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.

\override Beam.grow-direction = #LEFT
\featherDurations #(ly:make-moment 2/1)
{ c16[ 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.
Predefined commands
\featherDurations.

See also
Snippets: Section “Rhythms” in Snippets.

Known issues and warnings
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:

\bar

\[\text{e4 d c2} \ \bar{\text{"|."}}\]

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.

\[\text{Note: An incorrect duration can cause line breaks to be inhibited, leading to a line of highly compressed music or music which flows off the page.}\]

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 Section 4.3.1 [Line breaking], page 507.

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:
together with dotted and dashed bar lines:

```latex
\bar{;} \\
\bar{!} \\
al
```

and nine types of repeat bar lines:

```latex
\bar{.|:} \\
\bar{:..:} \\
\bar{:|.|:} \\
\bar{:|.:} \\
\bar{:.|.:} \\
\bar{[|:]} \\
\bar{:|][|:] \\
\bar{[:|]} \\
\bar{[:.|]} \\
al
```

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

```latex
\bar{\prime} \\
g
```

However, as such ticks are typically used in Gregorian chant, it is preferable to use `\divisioMinima` there instead, described in the section [Divisiones], page 418 in Gregorian chant.

Lilypond supports kievan notation and provides a special kievan bar line:
f1 \bar "k"

Further details of this notation are explained in Section 2.9.5 [Typesetting Kievan square notation], page 427.

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

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-S"
c4 c c c \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 Section 1.4 [Repeats], page 137), 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.
For combinations of repeats with the segno sign, there are six different variations:

For combinations of repeats with the segno sign, there are six different variations:
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], page 138.

New bar line types can be defined with \defineBarLine:

\defineBarLine barte

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 definition, the new bar line can be used by \bar barte.

There are currently ten bar line elements available:

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

\new Staff {
   s1 \bar "|"
   s1 \bar "."
   s1 \bar ":"
   s1 \bar ":"
   s1 \bar "!"
   s1 \bar ":"
   s1 \bar ":"
   s1 \bar "k"
   s1 \bar "S"
   s1 \bar ":=
   s1 \bar "="
   s1 \bar "="
   s1 \bar "["
   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.
Furthermore, the space character " " serves as a placeholder for defining span bars correctly aligned to the main bar lines:

```lilypond
\defineBarLine "||-dashedSpan" #'("||" "!"")
\new StaffGroup <<
  \new Staff {
    c1 \bar "||"
    c1 \bar "||-dashedSpan"
    c1
  }
  \new Staff {
    c1
    c1
    c1
  }
>>
```

![Musical notation example](image)

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

```lilypond
\defineBarLine ":.-wrong" #'(":." "|.")
\defineBarLine ":.-right" #'(":." " "|.")
\new StaffGroup <<
  \new Staff {
    c1 \bar ":.-wrong"
    c1 \bar ":.-right"
    c1
  }
  \new Staff {
    c1
    c1
    c1
  }
>>
```

![Musical notation example](image)

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

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

See also

Notation Reference: Section 4.3.1 [Line breaking], page 507, Section 1.4 [Repeats], page 137, [Grouping staves], page 176.

Installed Files: ‘scm/bar-line.scm’.

Snippets: Section “Rhythms” in Snippets.

Internals Reference: Section “BarLine” in Internals Reference (created at Staff level), Section “SpanBar” in Internals Reference (across staves), Section “Timing_translator” in Internals Reference (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:

```
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 \Score.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:

\override \Score.BarNumber.break-visibility = ##(#t #t #t)
\set \Score.currentBarNumber = #11
% Permit first bar number to be printed
\bar ""
\relative c' {
  \set \Score.barNumberVisibility = #all-bar-numbers-visible
  \bar ""
  c1 | d | e | f \break
  g1 | e | d | c
}\n
Selected Snippets

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 \Score.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.

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

Bar numbers can be printed at regular intervals by setting the property \Score.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 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)
    \c | c
    \% Center-align bar numbers
    \override Score.BarNumber.self-alignment-X = #CENTER
    \c | c
    \% Left-align bar numbers
    \override Score.BarNumber.self-alignment-X = #LEFT
    \c | c
}

\relative c'' {\break}
\relative c'' {\break}
\relative c'' {\break}
\relative c'' {\break}

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
        \remove "Bar_number_engraver"
    }
}

\relative c' {\break}
\relative c' {\break}
\relative c' {\break}
\relative c' {\break}

See also

Snippets: Section “Rhythms” in Snippets.

Internals Reference: Section “BarNumber” in Internals Reference,
Known issues and warnings

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 Section “StaffGroup” in Internals Reference and Section “BarNumber” in Internals Reference 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 |

Bar checks can also be used in lyrics:

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

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.

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.

See also

Snippets: Section “Rhythms” in Snippets.

Rehearsal marks

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

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).

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.
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

\mark \markup{ \musicglyph #"scripts.segno" }
\mark \markup{ \musicglyph #"scripts.coda" }
\mark \markup{ \musicglyph #"scripts.ufermata" }

See Section A.8 [The Feta font], page 624, for a list of symbols which may be printed with \musicglyph.

For common tweaks to the positioning of rehearsal marks, see Section 1.8.2 [Formatting text], page 223. For more precise control, see \break-alignable-interface in Section 5.5.1 [Aligning objects], page 591.

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.

See also

Notation Reference: Section A.8 [The Feta font], page 624, Section 1.8.2 [Formatting text], page 223, Section 5.5.1 [Aligning objects], page 591.

Installed Files: ‘scm/translation-functions.scm’.

Snippets: Section “Rhythms” in Snippets.

Internals Reference: Section “MarkEvent” in Internals Reference, Section “Mark_engraver” in Internals Reference, Section “RehearsalMark” in Internals Reference.

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.

\[ \text{c4} \ \text{\textbackslash grace} \ b16 \ a4( \ \text{\textbackslash grace} \ \{ \ b16 \ c16 \ } \ a2) \]

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 \texttt{\slashedGrace} function.

\[ \text{\textbackslash acciaccatura} \ d8 \ c4 \]
\[ \text{\textbackslash appoggiatura} \ e8 \ d4 \]
\[ \text{\textbackslash acciaccatura} \ \{ \ g16 \ f \ } \ e2 \]
\[ \text{\slashedGrace} \ a,8 \ g4 \]
\[ \text{\slashedGrace} \ b16 \ a4( \ \text{\slashedGrace} \ b8 \ a2) \]

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

\[ \langle \langle \text{\textbackslash new Staff} \ \{ \ e2 \ \text{\textbackslash grace} \ \{ \ c16 \ d \ e \ f \ } \ e2 \ \} \ \\
\text{\textbackslash new Staff} \ \{ \ c2 \ \text{\textbackslash grace} \ \{ \ g8 \ b \ } \ c2 \ \} \rangle \]

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

\[ \text{\textbackslash afterGrace} \ d1 \ \{ \ c16[ \ d \} \ c1 \]

This will put the grace notes after a space lasting 3/4 of the length of the main note. The default fraction 3/4 can be changed by setting \texttt{afterGraceFraction}. The following example shows the results from setting the space at the default, at 15/16, and finally at 1/2 of the main note.
Chapter 1: Musical notation

The space between the main note and the grace note may also be specified using spacers. The following example places the grace note after a space lasting 7/8 of the main note.

\new Voice {
<<
{ d1\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 {
\acciaccatura {
\stemDown 
f16->
\stemNeutral
}
g4 e c2
}
Selected Snippets

*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
    <>
}
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' {  
   \acciaccatura d8 c1  
}  
\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  
}  
}  

See also


Notation Reference: [Scaling durations], page 48, [Manual beams], page 86.

Installed Files: ’ly/grace-init.ly’.

Snippets: Section “Rhythms” in Snippets.

Internals Reference: Section “GraceMusic” in Internals Reference, Section “Grace_beam_engraver” in Internals Reference, Section “Grace_auto_beam_engraver” in Internals Reference, Section “Grace_engraver” in Internals Reference, Section “Grace_spacing_engraver” in Internals Reference.
Known issues and warnings

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,

\[
\begin{align*}
\text{new Staff} & \{ e4 \text{ bar } ".:1" \text{ grace c16 d2. } \} \\
\text{new Staff} & \{ c4 \text{ bar } ".:1" \text{ grace s16 d2. } \}
\end{align*}
\]

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

\[
\begin{align*}
\text{new Staff} & \{ e4 \text{ bar } ".:1" \text{ grace c16 d2. } \} \\
\text{new Staff} & \{ c4 \text{ bar } ".:1" \text{ grace s16 d2. } \}
\end{align*}
\]

The use of grace notes within voice contexts confuses the way the voice is typeset. This can be overcome by inserting a rest or note between the voice command and the grace note.

\[
\text{accMusic} = \{
\acciaccatura \{ f8 \} e8 r8 \acciaccatura \{ f8 \} e8 r4
\}
\]

\[
\text{new Staff} \{
\begin{align*}
\text{new Voice} & \{ \\
\text{relative } c' & \{ \\
\text{r8 r8 } \text{voiceOne } & \text{accMusic } \text{oneVoice r8 } | \\
\text{r8 } & \text{voiceOne } \text{r8 } \text{accMusic } \text{oneVoice r8 } | \\
\text{\}} \\
\text{\}} \\
\text{new Voice} & \{ \\
\text{relative } c' & \{ \\
\text{s8 s8 } & \text{voiceTwo } \text{accMusic } \text{oneVoice s8 } | \\
\text{s8 } & \text{voiceTwo } \text{r8 } \text{accMusic } \text{oneVoice s8 } | \\
\end{align*}
\]

Grace sections should only be used within sequential music expressions. Nesting or juxta-
posing 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:

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

becomes:

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

Or explicitly change the musical duration:

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

See [Scaling durations], page 48.

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 \texttt{mmrest-of-length} and \texttt{skip-of-length}. These Scheme functions take a defined piece of music as an argument and generate a multi-
measure rest or \texttt{\skip} exactly as long as the piece.

\texttt{MyCadenza = \relative c' \{ c4 d8 e f g g4 f2 g4 g \}}

\texttt{\new GrandStaff << \new Staff \{ \MyCadenza c'1 \MyCadenza c'1 \} \new Staff \{ #(mmrest-of-length MyCadenza) c'1 #(skip-of-length MyCadenza) c'1 \} \>>>
See also

Snippets: Section “Rhythms” in Snippets.

Time administration

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

The following properties of \texttt{Timing} are used to keep track of timing within the score.

- \texttt{currentBarNumber}
  - The current measure number. For an example showing the use of this property see [Bar numbers], page 96.

- \texttt{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.

- \texttt{measurePosition}
  - The point within the measure where we currently are. This quantity is reset by subtracting \texttt{measureLength} whenever \texttt{measureLength} is reached or exceeded. When that happens, \texttt{currentBarNumber} is incremented.

- \texttt{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 \texttt{measureLength} is set to 5/4. At 4/8 through the third measure, the \texttt{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.

\begin{verbatim}
\new Voice \relative c' {
    \set Timing.measureLength = #(ly:make-moment 5/4)
    c1 c4 |
    c1 c4 |
    c4 c
    \set Timing.measurePosition = #(ly:make-moment 5/8)
    b4 b b8 |
    c4 c1 |
}
\end{verbatim}

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

\texttt{note\name}

The possible values for \texttt{name} are listed in Section A.13 [List of articulations], page 699. For example:

\texttt{c4\staccato c\mordent b2\turn}
\texttt{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:

\begin{verbatim}
c4-- c-> c-- c--
c4-> c-. c2--
\end{verbatim}

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 Section 5.4.2 [Direction and placement], page 577.

Articulations are Script objects. Their properties are described more fully in Section “Script” in Internals Reference.

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.

\begin{verbatim}
\override Script.color = #red
\override MultiMeasureRestText.color = #blue
a2\fermata r\fermata
R1\fermataMarkup
\end{verbatim}

In addition to articulations, text and markups can be attached to notes. See [Text scripts], page 216.

For more information about the ordering of Scripts and TextScripts that are attached to the notes, see Section “Placement of objects” in Learning Manual.

**Selected Snippets**

*Modifying default values for articulation shorthand notation*

The shorthands are defined in `ly/script-init.ly`, where the variables dashHat, dashPlus, dashDash, dashBang, 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:

\begin{verbatim}
relative c' { c1-+ }
dashPlus = "trill"
relative c' { c1-+ }
\end{verbatim}
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 \\
    } \\
    >> 
}
See also


Notation Reference: [Text scripts], page 216, Section 5.4.2 [Direction and placement], page 577, Section A.13 [List of articulations], page 699, [Trills], page 135.

Installed Files: ‘scm/script.scm’.

Snippets: Section “Expressive marks” in Snippets.

Internals Reference: Section “Script” in Internals Reference, Section “TextScript” in Internals Reference.

Dynamics

Absolute dynamic marks are specified using a command after a note, such as c4\ff. The available dynamic marks are \pppp, \ppp, \pp, \p, \mp, \mf, \f, \ff, \fff, \ffff, \fp, \sf, \sp, \sp, \sp, \sp, \sfz, and \rfz. Dynamic marks may be manually placed above or below the staff; see Section 5.4.2 [Direction and placement], page 577.

\[c2\ppp c\mp c2\rfz c^\mf c2_\spp c^\ff\]

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.

\[c2\< c\! d2\< d\f e2\< e\> f2\> f\! e2\> e\mp d2\> d\> c1\!\]

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.

\[c1\< | c4 a c\< a | c4 a c\! a\< | c4 a c a\!\]
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.

\[c1< \quad \text{c4 a c\textcolor{red}{mf} a} \quad \text{c1<} \quad \text{c4 a c\textcolor{blue}{fff} a}\]

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:

\[c4\! \quad \text{c\textcolor{red}{!} d\textcolor{red}{>} e\textcolor{red}{!}} \quad \text{<< f1 \{ s4 s4\! s4\textcolor{red}{>} s4\textcolor{red}{!}\} >>}\]

The \textcolor{red}{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.

\[c2 \textcolor{red}{b4} \quad \textcolor{red}{a} \quad \textcolor{red}{g1\textcolor{red}{espressivo}}\]

Textual crescendo marks begin with \texttt{\cresc}. Textual decrescendos begin with \texttt{\decresc} or \texttt{\dim}. Extender lines are engraved as required.

\[g8\texttt{\cresc a b c b c d e\textcolor{red}{mf}} \quad f8\texttt{\decresc e d c e\textcolor{red}{>}} \quad \texttt{a1\dim \quad a2. \textcolor{red}{r4\textcolor{red}{!}}}\]

Textual marks for dynamic changes can also replace hairpins:

\texttt{\crescTextCresc} \texttt{\crescHairpin} \texttt{\dimTextDecr} \texttt{\dimTextDim}
Chapter 1: Musical notation

To create new absolute dynamic marks or text that should be aligned with dynamics, see [New dynamic marks], page 119.

Vertical positioning of dynamics is handled by Section “DynamicLineSpanner” in Internals Reference.

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.

Predefined commands
\dynamicUp, \dynamicDown, \dynamicNeutral, \crescTextCresc, \dimTextDim, \dimTextDecr, \dimTextDecresc, \crescHairpin, \dimHairpin.

Selected Snippets

\relative c' {\dimHairpin\c4\< d\! e\> d\! |}

\relative c' {c2 d4 e | c4 e e,2 | g'4 a g a | c1 |}
\new Staff \relative c' {s1\< | s1\f | s2\dim s2-"rit." | s1\p |}

\override Hairpin.to-barline = ##f
\relative c' {e4\< e2. e1\!}
\relative c' {e4\< e2.
e1\!}

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.
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.

\begin{verbatim}
relative c' {
    c4\< c! d\> e!
    \override Hairpin.minimum-length = #5
    << f1 { s4 s\< s\> s\! } >>
}
\end{verbatim}

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`.

\begin{verbatim}
relative c' {
    \override Hairpin.circled-tip = ##t
    c2\< c! 
    c4\> c< c2! 
}
\end{verbatim}

Printing hairpins in various styles

Hairpin dynamics may be created in a variety of styles

\begin{verbatim}
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!
}
\end{verbatim}
Vertically aligned dynamics and textscripts

All \textstyle \textsf{DynamicLineSpanner} objects (hairpins and dynamic texts) are placed with their reference line at least \textquoteleft staff-padding \textquoteleft from the staff, unless other notation forces them to be farther. Setting \textquoteleft staff-padding \textquoteleft to a sufficiently large value aligns the dynamics.

The same idea, together with \textquoteleft\textsf{textLengthOn}\textquoteleft, is used to align the text scripts along their baseline.

\begin{verbatim}
\markup \vspace #1 %avoid LSR-bug
\markup \textstyle music = \relative c' {
    a'2\p b\f
e4\p f\f\> g, b\p
c2\markup { \huge gorgeous } c\markup { \huge fantastic }
}
\markup \\break
\override DynamicLineSpanner.staff-padding = #3
\textLengthOn
\override TextScript.staff-padding = #1
\markup \textstyle music
\end{verbatim}

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:
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\mf  
}  

\relative c' {  
  \set crescendoText = \markup { \italic { cresc. poco } }  
  \set crescendoSpanner = #'text  
  \override DynamicTextSpanner.style = #'dotted-line  
  a2\< a  
  a2 a  
  a2 a\mf  
}  

See also


Notation Reference: Section 5.4.2 [Direction and placement], page 577, [New dynamic marks], page 119, Section 3.5.3 [What goes into the MIDI output?], page 486, Section 3.5.5 [Controlling MIDI dynamics], page 487.

Snippets: Section “Expressive marks” in Snippets.


New dynamic marks

The easiest way to create dynamic indications is to use `\markup` objects.

\moltoF = \markup { molto \dynamic f }

\relative c' {  
  <d e>16\_\moltoF <d e>
In markup mode, editorial dynamics (within parentheses or square brackets) can be created.
The syntax for markup mode is described in Section 1.8.2 [Formatting text], page 223.

\roundF = \markup { \center-align \concat { \normal-text \bold \italic ( } \dynamic f \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)
The Scheme form of markup mode may be used instead. Its syntax is explained in Section “Markup construction in Scheme” in Extending.

\relative c' {
    \timesig C44 \clef G
    \(\text{f}\)
    \(\text{f}\)
    \(\text{f}\)
    \(\text{f}\)
}

\relative c' {
    \timesig C44 \clef G
    \(\text{f}\)
    \(\text{f}\)
    \(\text{f}\)
    \(\text{f}\)
}

To left-align the dynamic text rather than centering it on a note use a \tweak:

\moltoF = \tweak DynamicText.self-alignment-X #LEFT
\relative c' {
    \timesig C44 \clef G
    \(\text{f}\)
    \(\text{f}\)
    \(\text{f}\)
    \(\text{f}\)
}

Font settings in markup mode are described in [Selecting font and font size], page 224.

See also

Notation Reference: Section 1.8.2 [Formatting text], page 223, [Selecting font and font size], page 224, Section 3.5.3 [What goes into the MIDI output?], page 486, Section 3.5.5 [Controlling MIDI dynamics], page 487.

Extending LilyPond: Section “Markup construction in Scheme” in Extending.

Snippets: Section “Expressive marks” in Snippets.
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:

```plaintext
\slurSolid {c4(e g2)}
\slurDotted {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):

```plaintext
\slurHalfDashed {c4(e g2)}
\slurHalfSolid {g4(e c2)}
```

Custom dash patterns for slurs can be defined:
Predefined commands
\slurUp, \slurDown, \slurNeutral, \slurDashed, \slurDotted, \slurHalfDashed, \slurHalfSolid, \slurDashPattern, \slurSolid.

Selected Snippets

Using double slurs for legato chords
Some composers write two slurs when they want legato chords. This can be achieved by setting doubleSlurs.
\relative c' {
\set doubleSlurs = ##t
\<c e>4( \<d f> \<c e> \<d f>)
}

Positioning text markups inside slurs
Text markups need to have the outside-staff-priority property set to false in order to be printed inside slurs.
\relative c'' {
\override TextScript.avoid-slur = #'inside
\override TextScript.outside-staff-priority = ##f
\c2( \markup { \halign #-10 \natural } d4.) c8
}

Making slurs with complex dash structure
Slurs can be made with complex dash patterns by defining the dash-definition property. dash-definition is a list of dash-elements. A dash-element is a list of parameters defining the dash behavior for a segment of the slur.

The slur is defined in terms of the bezier parameter t which ranges from 0 at the left end of the slur to 1 at the right end of the slur. dash-element is a list (start-t stop-t...}
dash-fraction dash-period). The region of the slur from start-t to stop-t will have a fraction dash-fraction of each dash-period black. dash-period is defined in terms of staff spaces. dash-fraction is set to 1 for a solid slur.

\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)
}

See also
Notation Reference: Section 5.4.2 [Direction and placement], page 577, [Phrasing slurs], page 124.
Snippets: Section “Expressive marks” in Snippets.
Internals Reference: Section “Slur” in Internals Reference.

Phrasing slurs
Phrasing slurs (or phrasing marks) that indicate a musical sentence are written using the commands \( and \) respectively:
\( c4\( d( e) f( e2) d\)\)

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 Section 5.4.2 [Direction and placement], page 577.

Simultaneous or overlapping phrasing slurs are not permitted.
Phrasing slurs can be solid, dotted, or dashed. Solid is the default style for phrasing slurs:
\( c4\( e g2\)\)
\phrasingSlurDashed
\( g4\( e c2\)\)
\phrasingSlurDotted
\( c4\( e g2\)\)
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):

\begin{verbatim}
c4\( e \ g2\)
\phrasingSlurHalfDashed
g4\( e \ c2\)
\phrasingSlurHalfSolid
c4\( e \ g2\)
\phrasingSlurSolid
g4\( e \ c2\)
\end{verbatim}

Custom dash patterns for phrasing slurs can be defined:

\begin{verbatim}
c4\( e \ g2\)
\phrasingSlurDashPattern #0.7 #0.75
g4\( e \ c2\)
\phrasingSlurDashPattern #0.5 #2.0
c4\( e \ g2\)
\phasingSlurSolid
g4\( e \ c2\)
\end{verbatim}

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], page 122.

**Predefined commands**

\phrasingSlurUp, \phrasingSlurDown, \phrasingSlurNeutral, \phrasingSlurDashed, \phrasingSlurDotted, \phrasingSlurHalfDashed, \phrasingSlurHalfSolid, \phrasingSlurDashPattern, \phrasingSlurSolid.

**See also**

Notation Reference: Section 5.4.2 [Direction and placement], page 577, [Slurs], page 122.
Snippets: Section “Expressive marks” in Snippets.
Internals Reference: Section “PhrasingSlur” in Internals Reference.
Breath marks

Breath marks are entered using \breathe:

\c2 \breathe d4

\c8 \breathe d e f g2

A breath mark will end an automatic beam; to override this behavior, see [Manual beams], page 86.

Musical indicators for breath marks in ancient notation, divisiones, are supported. For details, see [Divisiones], page 418.

Selected Snippets

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

See also

Notation Reference: [Divisiones], page 418.
Snippets: Section “Expressive marks” in Snippets.

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.

c2\bendAfter +%4
c2\bendAfter +%4
c2\bendAfter +%6.5
c2\bendAfter +%6.5
c2\bendAfter +%8
c2\bendAfter +%8
Selected Snippets

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 
}
```

See also


Snippets: Section “Expressive marks” in Snippets.

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:

```
g2\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'' {
\time 3/4
\override Glissando.style = #'zigzag
c4 c
\cadenzaOn
c4\glissando
\hideNotes
c,,4
\unHideNotes
\cadenzaOff
\bar "|"
}
```

Different styles of glissandi can be created. For details, see Section 5.4.7 [Line styles], page 590.

**Selected Snippets**

*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
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.

\begin{verbatim}
repeatGliss = #(define-music-function (parser location 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
  #})
\end{verbatim}

\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 e1 }
\}
}

\score {
\new StaffGroup <<
\new Staff <<
\context Voice { \clef "G_8" \music }
>>
\new TabStaff <<
\context TabVoice { \clef "moderntab" \music }
>>
>>
}

See also

Notation Reference: Section 5.4.7 [Line styles], page 590.
Snippets: Section “Expressive marks” in Snippets.
Internals Reference: Section “Glissando” in Internals Reference.

Known issues and warnings

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:

```
<cegc>1\arpeggio
```

Different types of arpeggios may be written. `\arpeggioNormal` reverts to a normal arpeggio:

```
<cegc>2\arpeggio
```

```
\arpeggioArrowUp
<cegc>2\arpeggio
```

```
\arpeggioArrowDown
<cegc>2\arpeggio
```

```
\arpeggioNormal
<cegc>2\arpeggio
```

Special *bracketed* arpeggio symbols can be created:

```
<cegc>2
```

```
\arpeggioBracket
<cegc>2\arpeggio
```

```
\arpeggioParenthesis
<cegc>2\arpeggio
```

```
\arpeggioParenthesisDashed
<cegc>2\arpeggio
```

```
\arpeggioNormal
<cegc>2\arpeggio
```

The dash properties of the parenthesis arpeggio are controlled with the `\dash-details` property, which is described at [Slurs], page 122.

Arpeggios can be explicitly written out with ties. For more information, see [Ties], page 49.

Predefined commands

`\arpeggio`, `\arpeggioArrowUp`, `\arpeggioArrowDown`, `\arpeggioNormal`, `\arpeggioBracket`, `\arpeggioParenthesis` `\arpeggioParenthesisDashed`.
Selected Snippets

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
    }
  }
}
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 }
  \>>
}

Known issues and warnings

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], page 307.

Trills

Short trills without an extender line are printed with \trill; see [Articulations and ornamentations], page 111.

Longer trills with an extender line are made with \startTrillSpan and \stopTrillSpan:
A trill spanner crossing a line break will restart exactly above the first note on the new line.

Consecutive trill spans will work without explicit \stopTrillSpan commands, since successive trill spanners will automatically become the right bound of the previous trill.

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], page 104.

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.
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.

Predefined commands
\startTrillSpan, \stopTrillSpan.

See also
Notation Reference: [Articulations and ornamentations], page 111, [Grace notes], page 104.
Snippets: Section “Expressive marks” in Snippets.
Internals Reference: Section “TrillSpanner” in Internals Reference.

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:

```
\repeat volta 2 \{ c4 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.
where \texttt{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:
\begin{verbatim}
\repeat volta 2 { c4 d e f | }
\alternative {
  { c2 e | }
  { f2 g | }
}
\end{verbatim}

Multiple repeats with one alternate ending:
\begin{verbatim}
\repeat volta 4 { c4 d e f | }
\alternative {
  { c2 e | }
  { f2 g | }
}
\end{verbatim}

Multiple repeats with more than one alternate ending:
\begin{verbatim}
\repeat volta 3 { c4 d e f | }
\alternative {
  { c2 e | }
  { f2 g | }
  { a2 g | }
}
\end{verbatim}

\textbf{Note:} 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 \texttt{alternative} block, otherwise you will not get the expected number of endings.
If a repeat starts in the middle of a measure and has no alternate endings, normally the end of the repeat will also fall in the middle of a measure, so that the two ends add up to one complete measure. In such cases, the repeat signs do not constitute true bar lines. Do not use \partial commands or bar checks where these repeat signs are printed:

\begin{verbatim}
% no \partial here
c4 e g  % no bar check here
% no \partial here
\repeat volta 4 {
  e4 |
c2 e |
  % no \partial here
g4 g g  % no bar check here
}  
% no \partial here
g4 |
a2 a |
g1 |
\end{verbatim}

Similarly, if a repeat begins with the initial partial measure of a score and has no alternate endings, the same conditions apply as in the above example, except that in this case the \partial command is required at the beginning of the score:

\begin{verbatim}
\partial 4  % required
\repeat volta 4 {
  e4 |
c2 e |
  % no \partial here
g4 g g  % no bar check here
}  
% no \partial here
g4 |
a2 a |
g1 |
\end{verbatim}

When alternate endings are added to a repeat that begins with an incomplete measure, it becomes necessary to set the Timing.measureLength context property manually, in the following specific places:

- at the start of any incomplete measures in the \alternative block, which normally occur at the end of each alternative, except (in most cases) the last.
- at the start of each alternative, except the first.
\partial 4
\repeat volta 2 { e4 | c2 e | }  
\alternative {
{  
f2 d |
\set Timing.measureLength = #(ly:make-moment 3/4)
\set Staff-layout overrides = #(ly:make-moment 3/4)
g4 g g  \% optional bar check is allowed here 
}
{  
\set Timing.measureLength = #(ly:make-moment 4/4)
a2 a |
}
}  
g1 | 

The \texttt{measureLength} property is described in \cite{Time administration}, page 110.

Ties may be added to a second ending:
\begin{verbatim}
c1
\repeat volta 2 { c4 d e f~ }  
\alternative {
{  
f2 d }
{  
f2\repeatTie f, }
}
\end{verbatim}

The \texttt{\textbackslash 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 \texttt{\textbackslash 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:
\begin{verbatim}
e1
\inStaffSegno
f2 g a b  
c1_"D.S." \bar "|.
\end{verbatim}

At the start of a repeat:
\begin{verbatim}
e1
\repeat volta 2 {  
\inStaffSegno  \% start repeat
\end{verbatim}
At the end of a repeat:
\\r
\repeat volta 2 {
  f2 g a b
  \inStaffSegno % end repeat
}
f2 g a b
c1_"D.S." \bar "."

Between two repeats:
\\r
\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 \texttt{segnoType}, \texttt{startRepeatSegnoType}, \texttt{endRepeatSegnoType} or \texttt{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 \texttt{\defineBarLine} command (see [Bar lines], page 90).
\defineBarLine ":|.S[" ":.|." "S[" ""
\defineBarLine "]" ":|" ""
\defineBarLine "]" ":|" ""
\defineBarLine ":|.S[" ":.|." "S[" ""
\defineBarLine e1
\repeat volta 2 {
  f2 g a b
  \once \set Score.endRepeatSegnoType = ":|.S[
    \inStaffSegno
  }
  f2 g \bar "]" a b
c1_"D.S." \bar "."
Selected Snippets

*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`.

```latex
\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 }
    \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.

```latex
\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
}
```
See also


Notation Reference: [Bar lines], page 90, Section 5.1.4 [Modifying context plug-ins], page 553, [Modifying ties and slurs], page 597, [Time administration], page 110.

Snippets: Section “Repeats” in Snippets.


Known issues and warnings

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], page 597.

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], page 128.

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
Note: 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], page 90.

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.

\begin{verbatim}
c1\set Score.repeatCommands = #'(start-repeat)
d4 e f g
c1
\end{verbatim}

As per standard engraving practice, repeat signs are not printed at the beginning of a piece.

**end-repeat**

Print a `:|.` bar line:

\begin{verbatim}
c1
d4 e f g
\set Score.repeatCommands = #'(end-repeat)
c1
\end{verbatim}

\begin{verbatim}
(folta number) ... (volta #f)
\end{verbatim}

Create a new volta with the specified number. The volta bracket must be explicitly terminated, or it will not be printed.

\begin{verbatim}
f4 g a b\set Score.repeatCommands = #'(volta "2")
g4 a g a\set Score.repeatCommands = #'(volta #f)
c1
\end{verbatim}

Multiple repeat commands may occur at the same point:

\begin{verbatim}
f4 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
\end{verbatim}
\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 Section 1.8.2 [Formatting text], page 223. The simplest way to use markup text is to define the markup first, then include the markup in a Scheme list.

\begin{verbatim}
voltaAdLib = \markup { 1. 2. 3... \text \italic { ad lib. } }
\relative c'' {
  c1
  \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)
}
\end{verbatim}

Selected Snippets

Printing a repeat sign at the beginning of a piece

A .|: bar line can be printed at the beginning of a piece, by overriding the relevant property:

\begin{verbatim}
\relative c'' {
  \once \override Score.BreakAlignment.break-align-orders = #(make-vector 3 '(instrument-name
    left-edge
    ambitus
    breathing-sign
    clef
    key-signature
    time-signature
    staff-bar
    custos))

  \once \override Staff.TimeSignature.space-alist = #'((first-note . (fixed-space . 2.0))
    (right-edge . (extra-space . 0.5))
    ;; free up some space between time signature
    ;; and repeat bar line
    (staff-bar . (extra-space . 1)))

  \bar ".|:"
  c1
d1
d4 e f g
}
\end{verbatim}
See also

Notation Reference: [Bar lines], page 90, Section 1.8.2 [Formatting text], page 223.

Snippets: Section “Repeats” in Snippets.


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.

```
\repeat unfold 2 { c4 d e f }
```

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.

```
\repeat unfold 2 { c4 d e f }
\alternative {
    { c2 g' }
    { c,2 b }
}
```

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.

```
\repeat unfold 4 { c4 d e f }
\alternative {
    { c2 g' }
    { c,2 b }
    { e2 d }
}
```

If there are more alternate endings than repeats then only the first alternatives are applied. The remaining alternatives will be ignored and not printed.

\repeat unfold 2 { c4 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).

\repeat unfold 2 {
  \repeat unfold 2 { c4 d e f }
  \alternative {
    \{ c2 g' \}
    \{ c,2 b \}
  }
}
c1

Chord constructs can be repeated by the chord repetition symbol q. See [Chord repetition], page 156.

**Note:** If you include \relative inside a \repeat without explicitly instantiating the Voice context, extra (unwanted) staves will appear. See Section “An extra staff appears” in Application Usage.

See also

Notation Reference: [Chord repetition], page 156.
Snippets: Section “Repeats” in Snippets.
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.

\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.

\repeat percent 2 { c4 d e f }
\repeat percent 2 { c2 d }
\repeat percent 2 { c1 }

Patterns that are shorter than one measure but contain mixed durations use a double-percent symbol.

\repeat percent 4 { c8. <d f>16 }
\repeat percent 2 { \tuplet 3/2 { r8 c d } e4 }
Selected Snippets

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 (parser location note) (ly:music?) "Make a percent repeat the same length as NOTE."
        (make-music 'PercentEvent
            'length (ly:music-length note)))

\relative c'' {
    \makePercent s1
}

\relative c'' {
    \makePercent s1
}

\relative c'' {
    \makePercent s1
}
See also


Snippets: Section “Repeats” in Snippets.

Internals Reference: Section “RepeatSlash” in Internals Reference, Section “Repeat-SlashEvent” in Internals Reference, Section “DoubleRepeatSlash” in Internals Reference, Section “PercentRepeat” in Internals Reference, Section “PercentRepeatCounter” in Internals Reference, Section “Percent_repeat_ engraver” in Internals Reference, Section “DoublePercentEvent” in Internals Reference, Section “DoublePercentRepeat” in Internals Reference, Section “DoublePercentRepeatCounter” in Internals Reference, Section “Double_percent_repeat_ engraver” in Internals Reference, Section “Slash_repeat_ engraver” in Internals Reference.

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:

\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 (stored in tremoloFlags) is used:

c2:8 c:32
c: c:

Selected Snippets

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
      }
    }
  }
>>

See also

Snippets: Section “Repeats” in Snippets.

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.

\[
\text{<a c e>1 <a c e>2 <f a c e>4 <a c>8. <g c e>16}
\]

Chords may also be followed by articulations, again just like simple notes.

\[
\text{<a c e>1\fermata <a c e>2-> <f a c e>4\prall <a c>8.}! <g c e>16^-.
\]

The notes within the chord themselves can also be followed by articulation and ornamentation.

\[
\text{<a c\prall e>1 <a-> c-- e>2 <f-- a c-- e-->4 <a-- c-->8. <g\fermata c e\turn>16}
\]
However some notation, such as dynamics, hairpins and slurs must be attached to the chord, rather than notes within the chord, otherwise they will not print.

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], page 157):

```
\grace { g8( 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.

```
<a c e>1 <f a c> <a c e> <f' a c> <b, e b,>
```

For more information about chords, see Section 2.7 [Chord notation], page 384.

**See also**


Notation Reference: Section 2.7 [Chord notation], page 384, [Articulations and ornamentations], page 111, [Relative octave entry], page 2, Section 1.5.2 [Multiple voices], page 159.

Snippets: Section “Simultaneous notes” in *Snippets*.

**Known issues and warnings**

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 Section 1.5.2 [Multiple voices], page 159, ‘<< f! \ < e fis!> >>’,
- enharmonic transcription of one or more pitches, ‘<e f ges>’, or
- [Clusters], page 158.
Chord repetition

In order to save typing, a shortcut can be used to repeat the preceding chord. The chord repetition symbol is q:

\[ \text{c e a} \] \text{1 q} \text{f a c} \text{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.

\[ \text{c e a} \] \text{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.

\[ \text{c e a} \] \text{1 c'4 q2 r8 q8 |} \\
\text{q2 c, |} \\

However, the chord repetition symbol does not retain any dynamics, articulation or ornamentation within, or attached to, the previous chord.

\[ \text{c e a} \] \text{1 \prall e \sfz c'4 q2 r8 q8 |} \\
\text{q2 c, |} \\

To have some of them retained, the \texttt{\chordRepeats} function can be 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.

\[ \relative c' \{ \chordRepeats #'(articulation-event) \{ <a-. c\prall e \sfz c'4 q2 r8 q8-. } | \\
\text{q2 c, |} \\
\} \\
\]
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.

\new Voice
\relative c'' {
  \chordRepeats #'(articulation-event)
  \relative c''
  { <a-. c\prall e>1\sfz 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.

See also

Notation Reference: Section 2.7 [Chord notation], page 384, [Articulations and ornamentations], page 111.


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
  << { a4 b g2 } { d4 g c,2 } >>
}

% single first note
a << { a4 b g } { d4 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:
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:

Warning: ignoring too many clashing note columns

Here different rhythms cause no problems because they are interpreted in different voices.

**Known issues and warnings**

If notes from two or more voices, with stems in the same direction, are placed at the same position on the staff and have no shift (or have the same shift specified), the message:

warning: ignoring too many clashing note columns

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, page 162).

**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.,

\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.

**See also**


Snippets: Section “Simultaneous notes” in Snippets.

Known issues and warnings

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"
{ \voicenumOne r8 r16 g e8. f16 g8[ c,] f e16 d }
\new Voice= "second"
{ \voicenumTwo d16 c d8~ d16 b c8~ c16 b c8~ c16 b8. }
>>
```

Here, voices are instantiated explicitly and are given names. The `\voicenumOne` ... `\voicenumFour` 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:

```
<< { \voicenumOne ... }
\new Voice { \voicenumTwo ... }
>> \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:

```
<<
\new Voice = "melody" {
  a4
  <<
  { \voicenumOne
    g f
  }
}
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:

\<<
{ r8 r16 g e8. f16 g8[ c,] f e16 d }
\{ d16 c d8~ d16 b c8~ c16 b c8~ c16 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 duplicated rests.

\<<
{ r8 g g g f16 ees f8 d }
\{ ees,8 r ees r d r d r }
\{ d'8 s c s bes s a s }
>>
In all but the simplest works it is advisable to create explicit \texttt{Voice} contexts as explained in Section “Contexts and engravers” in \textit{Learning Manual} and Section “Explicitly instantiating voices” in \textit{Learning Manual}.

\textbf{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:

\begin{verbatim}
\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
>>
\end{verbatim}

\begin{center}
\begin{music}
\clearpage
\end{music}
\end{center}

\textbf{Note:} Lyrics, spanners (such as slurs, ties, hairpins etc.) cannot be created 'across' voices.

\textbf{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 \texttt{Voice} context, thus forming chords. To achieve this, enclose them in a simple simultaneous music construct within an explicit voice:

\begin{verbatim}
\new Voice <<
  { e4 f8 d e16 f g8 d4 }
  { c4 d8 b c16 d e8 b4 }
>>
\end{verbatim}

\begin{center}
\begin{music}
\clearpage
\end{music}
\end{center}
This method leads to strange beamings and warnings if the pieces of music do not have the same rhythm.

**Predefined commands**

\voiceOne, \voiceTwo, \voiceThree, \voiceFour, \oneVoice.

**See also**


Notation Reference: [Percussion staves], page 363, [Invisible rests], page 54, [Stems], page 210.

Snippets: Section “Simultaneous notes” in *Snippets*.

**Voice styles**

Voices may be given distinct colors and shapes, allowing them to be easily identified:

```
<<
{ \voiceOneStyle d4 c2 b4 }
\\
{ \voiceTwoStyle e,2 e }
\\
{ \voiceThreeStyle b2. c4 }
\\
{ \voiceFourStyle g'2 g }
>>
```

The \voiceNeutralStyle command is used to revert to the standard presentation.

**Predefined commands**

\voiceOneStyle, \voiceTwoStyle, \voiceThreeStyle, \voiceFourStyle, \voiceNeutralStyle.

**See also**


Snippets: Section “Simultaneous notes” in *Snippets*.

**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.

```
<<
{ c8 d e d c d c4
```
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:

\begin{verbatim}
\mergeDifferentlyHeadedOn
\mergeDifferentlyDottedOn
c8 d e d c d c4
g'2 fis
\end{verbatim}

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 also be merged:

\begin{verbatim}
\mergeDifferentlyHeadedOn
\mergeDifferentlyDottedOn
c8 d e d c d c4
g'2 fis
\end{verbatim}
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.

```
<<
{\mergeDifferentlyHeadedOn \mergeDifferentlyDottedOn c8 d e d c d c4 \shiftOn g'2 fis}
\{ c2 c8. b16 c4 e,2 r \}
\{ \oneVoice \}
{ s1 e8 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.

```
Note: 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!
```
Two additional commands, \shiftOnn and \shiftOnnn provide further shift levels which may be specified temporarily to resolve collisions in complex situations – see Section “Real music example” in Learning Manual.

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).

**Predefined commands**

\mergeDifferentlyDottedOn, \mergeDifferentlyDottedOff, \mergeDifferentlyHeadedOn, \mergeDifferentlyHeadedOff.

\shiftOn, \shiftOnn, \shiftOnnn, \shiftOff.

**Selected Snippets**

*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.

```plaintext
voiceFive = #(context-spec-music (make-voice-props-set 4) 'Voice)
```

```plaintext
\relative c' { 
  \time 3/4
  \key d \minor
  \partial 2
  <<
    \new Voice { 
      \voiceOne
      a4. a8
  }
}
```
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>
}
\langle
{   <b f'>2 
    \once \override NoteColumn.force-hshift = \#1.7
    <b f'>2
}
See also


Snippets: Section “Simultaneous notes” in *Snippets*.

Internals Reference: Section “NoteColumn” in *Internals Reference*, Section “NoteCollision” in *Internals Reference*, Section “RestCollision” in *Internals Reference*.

Known issues and warnings

Using \override NoteColumn.ignore-collision = ##t will cause differently headed notes in different voices to merge incorrectly.

\mergeDifferentlyHeadedOn
<< { c16 a' b a } \ \ { c,2 } >>
\override NoteColumn.ignore-collision = ##t
<< { c16 a' b a } \ \ { c,2 } >>

\[ \text{\includegraphics{noteheads.s2}} \]

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.

\( \text{\texttt{instrumentOne = \relative c' {}}}
\begin{verbatim}
| c\texttt{4} d e f |
R\texttt{1} |
| d\texttt{4} c b a |
| b\texttt{4} g2 f\texttt{4} |
| e\texttt{1} |
\end{verbatim}
\)

\( \text{\texttt{instrumentTwo = \relative g' {}}}
\begin{verbatim}
| R\texttt{1} |
| g\texttt{4} a b c |
| d\texttt{4} c b a |
| g\texttt{4} f(e) d |
| e\texttt{1} |
\end{verbatim}
\)

\( << \)
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”.

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 the following commands:

Commands ending in \function{Once} apply only to the next note in the music expression.

- \partcombineApart and \partcombineApartOnce keep the notes as two separate voices, even if they can be combined into a chord or unison.
- \partcombineChords and \partcombineChordsOnce combine the notes into a chord.
- \partcombineUnisono and \partcombineUnisonoOnce combine both voices as “unison”.
- \partcombineSoloI and \partcombineSoloIOne print only voice one, and mark it as a “Solo”.
- \partcombineSoloII or \partcombineSoloIIOne print only voice two and mark it as a “Solo”.
- \partcombineAutomatic and \partcombineAutomaticOnce end the functions of the commands above, and revert back to the standard \partcombine functionality.

\begin{verbatim}
instrumentOne = \relative c' { 
  \partcombineApart c2"apart" e | 
  \partcombineAutomatic e2"auto" e | 
}\end{verbatim}
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 partcombiner will stop working. However, this effect can be achieved using a NullVoice context. See [Polyphony with shared lyrics], page 268.

Selected Snippets

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.

\begin{verbatim}
musicUp = \relative c'' { 
\time 4/4
\partcombineChords e'2^"chord" e |
\partcombineAutomatic c2^"auto" c |
\partcombineApart c2^"apart" \partcombineChordsOnce e^"chord once" |
c2 c |
}
instrumentTwo = \relative c' { 
c2 c |
e2 e |
a,2 c |
c2 c' |
c2 c |
c2 c |
}

<<
\new Staff { \instrumentOne }
\new Staff { \instrumentTwo }
\new Staff { \partcombine \instrumentOne \instrumentTwo }
>>
\end{verbatim}
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
  }
>>
```

```
girl boy together
```

See also

Notation Reference: Section 1.6.3 [Writing parts], page 192.
Snippets: Section “Simultaneous notes” in Snippets.

Known issues and warnings

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.
\textbf{\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 \textit{Known issues and warnings} when using \partcombine with tablature in [Default tablatures], page 318 and the \textit{Note} in [Automatic beams], page 76 when using automatic beaming.

\textbf{Writing music in parallel}

Music for multiple parts can be interleaved in input code. The function \texttt{\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.

\textbf{Note:} Bar checks | must be used, and the measures must be of the same length.

\begin{verbatim}
\parallelMusic #'(voiceA voiceB voiceC) { 
  % Bar 1
  r8 g'16 c'' e'' g' c'' e'' g' c'' e'' r8 g'16 c'' e'' g' c'' e'' |
  r16 e'8.~ e'4 r16 e'8.~ e'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.~ d'4 r16 d'8.~ d'4 |
  c'2 c'2 |
}
\new StaffGroup <<
  \new Staff << \voiceA \ \voiceB >>
  \new Staff { \clef bass \voiceC }
>>
\end{verbatim}

Relative mode may be used. Note that the \texttt{\relative} command is not used inside \texttt{\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.

\begin{verbatim}
\parallelMusic #'(voiceA voiceB voiceC) { 
  % Bar 1
  r8 g16 c e g, c e r8 g,16 c e g, c e |
  r16 e8.~ e4 r16 e8.~ e4 |
  c2 c |
\end{verbatim}
This works quite well for piano music. This example maps four consecutive measures to four variables:

```latex
\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 ...
}
```

```latex
\score {
\new PianoStaff <<
\new Staff {
\global <<
\relative c'' ! \voiceA
! ! \relative c' \voiceB
! ! ! \relative c' ! \voiceC
>>
}
```
See also

Snippets: Section “Simultaneous notes” in Snippets.

1.6 Staff notation

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 Section 5.1.2 [Creating and referencing contexts], page 547.

The basic staff context is Staff:

```
\new Staff { c4 d e f }
```

![Basic Staff Example](image)

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], page 363.

```
\new DrumStaff {
  \drummode { cymc hh ss tomh }
}
```

![Drum Staff Example](image)

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], page 73.

```
\new RhythmicStaff { c4 d e f }
```

![Rhythmic Staff Example](image)

TabStaff creates a tablature with six strings in standard guitar tuning. For details, see [Default tablatures], page 318.

```
\new TabStaff { c4 d e f }
```

![Tab Staff Example](image)

There are two staff contexts specific for the notation of ancient music: MensuralStaff and VaticanaStaff. They are described in [Pre-defined contexts], page 407.

The GregorianTranscriptionStaff context creates a staff to notate modern Gregorian chant. It does not show bar lines.

```
\new GregorianTranscriptionStaff { c4 d e f e d }
```

![Gregorian Staff Example](image)

New single staff contexts may be defined. For details, see Section 5.1.6 [Defining new contexts], page 560.
See also


Notation Reference: Section 5.1.2 [Creating and referencing contexts], page 547, [Percussion staves], page 363, [Showing melody rhythms], page 73, [Default tablatures], page 318, [Pre-defined contexts], page 407, [Staff symbol], page 182, [Gregorian chant contexts], page 416, [Mensural contexts], page 408, Section 5.1.6 [Defining new contexts], page 560.

Snippets: Section “Staff notation” in Snippets.

Internals Reference: Section “Staff” in Internals Reference, Section “DrumStaff” in Internals Reference, Section “GregorianTranscriptionStaff” in Internals Reference, Section “RhythmicStaff” in Internals Reference, Section “TabStaff” in Internals Reference, Section “MensuralStaff” in Internals Reference, Section “VaticanaStaff” in Internals Reference, Section “StaffSymbol” in Internals Reference.

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 { c1 c }
\new Staff { c1 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 { c1 c }
\new Staff { c1 c }
>>
```

In a ChoirStaff, the group starts with a bracket, but bar lines are not connected.

```
\new ChoirStaff <<
\new Staff { c1 c }
\new Staff { c1 c }
>>
```
In a GrandStaff, the group begins with a brace, and bar lines are connected between the staves.

```
\new GrandStaff <<
  \new Staff { c1 c }
  \new Staff { c1 c }
>>
```

The PianoStaff is identical to a GrandStaff, except that it supports printing the instrument name directly. For details, see [Instrument names], page 192.

```
\new PianoStaff <<
  \set PianoStaff.instrumentName = \"Piano\"
  \new Staff { c1 c }
  \new Staff { 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 Section 5.1.6 [Defining new contexts], page 560.

**Selected Snippets**

**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 by setting the transparent property.

global = {
    \hide Staff.BarLine
    s1 s
    % the final bar line is not interrupted
    \undo \hide Staff.BarLine
    \bar "|.
}
\new StaffGroup \relative c’\prime { 
  \new Staff { \<< \global { c1 c } \>> } 
  \new Staff { \<< \global { c c } \>> } 
  \} 
}

See also


Notation Reference: [Instrument names], page 192, Section 5.1.6 [Defining new contexts], page 560.

Snippets: Section “Staff notation” in Snippets.

Internals Reference: Section “Staff” in Internals Reference, Section “StaffGroup” in Internals Reference, Section “ChoirStaff” in Internals Reference, Section “GrandStaff” in Internals Reference, Section “PianoStaff” in Internals Reference, Section “SystemStartBar” in Internals Reference, Section “SystemStartBrace” in Internals Reference, Section “SystemStartBracket” in Internals Reference, Section “SystemStartSquare” in Internals Reference.

Known issues and warnings

PianoStaff does not, by default, accept ChordNames.

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 { c2 c \| c2 c } 
  \new StaffGroup << 
    \new Staff { g2 g \| g2 g } 
    \new StaffGroup \with { 
      systemStartDelimiter = #'SystemStartSquare 
    } 
  \new Staff { e2 e \| e2 e } 
  \new Staff { c2 c \| c2 c } 
  \} 
}
New nested staff group contexts can be defined. For details, see Section 5.1.6 [Defining new contexts], page 560.

Selected Snippets

Nesting staves

The property \texttt{systemStartDelimiterHierarchy} can be used to make more complex nested staff groups. The command \texttt{\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 \texttt{SystemStartBar}, \texttt{SystemStartBracket}, \texttt{SystemStartBrace}, and \texttt{SystemStartSquare}.

\begin{verbatim}
\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 }
\new Staff { c1 }
>>
\end{verbatim}
See also

Notation Reference: [Grouping staves], page 176, [Instrument names], page 192, Section 5.1.6 [Defining new contexts], page 560.

Snippets: Section “Staff notation” in Snippets.

Internals Reference: Section “StaffGroup” in Internals Reference, Section “ChoirStaff” in Internals Reference, Section “SystemStartBar” in Internals Reference, Section “SystemStartBrace” in Internals Reference, Section “SystemStartBracket” in Internals Reference, Section “SystemStartSquare” in Internals Reference.

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' {
        c4 c c c
        \break
        c4 c c c
      }
    }
    \new Staff {
      \relative c' {
        c4 c c c
        \break
        c4 c c c
      }
    }
    >>
  }
  \paper {
    system-separator-markup = \slashSeparator
    \%
  }
}
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.

\stopStaff f4 d \startStaff g, e
f'4 d \stopStaff g, e
f'4 d \startStaff g, e

Predefined commands

\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:
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.

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], page 16.

Staff line thickness can be altered. Ledger lines and note stems, by default, are also affected.

It is also possible to set ledger line thickness independently of staff lines.
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,
\begin{verbatim}
\new Staff \with {
    \override StaffSymbol.ledger-positions = #'(-3 -2 -1 2 5 6)
}
\{ f'4 a, a,, f \}
\end{verbatim}

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.
\begin{verbatim}
\new Staff \with {
    \override StaffSymbol.ledger-extra = #4
}
\{ f'4 a, d, f, \}
\end{verbatim}

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.
\begin{verbatim}
\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
\end{verbatim}

The distance between staff lines can be altered. This affects ledger line spacing as well.
\begin{verbatim}
\new Staff \with {
    \override StaffSymbol.staff-space = #1.5
}
\{ f'4 d, g, e, \}
\end{verbatim}
Selected Snippets

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'
}
```

See also

Music Glossary: Section “line” in Music Glossary, Section “ledger line” in Music Glossary, Section “staff” in Music Glossary.

Notation Reference: [Clef], page 16.

Snippets: Section “Staff notation” in Snippets.

Internals Reference: Section “StaffSymbol” in Internals Reference, Section “staff-symbol-interface” in Internals Reference.

Ossia staves

Ossia staves can be set by creating a new simultaneous staff in the appropriate location:

```
\new Staff \relative c'' { 
  c4 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 Section “Nesting music expressions” in Learning Manual.

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.
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
    \fontSize = #-3
    \override StaffSymbol.staff-space = #(magstep -3)
    \override StaffSymbol.thickness = #(magstep -3)
    firstClef = ##f
}
{ \stopStaff s1*6 }
\new Staff \relative c' {
    c4 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" {
    \new Staff = "main" \relative c' {
        c4 b d c
        <<
        { c4 b d c }
        \new Staff \with {
            \remove "Time_signature_engraver"
            alignAboveContext = "#main"
            \fontSize = #-3
            \override StaffSymbol.staff-space = #(magstep -3)
            \override StaffSymbol.thickness = #(magstep -3)
            firstClef = ##f
        }
        { e4 d f e }
    }}
    c4 b c2
}
Using the \Staff \RemoveEmptyStaves 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 \Staff \RemoveEmptyStaves, see [Hiding staves], page 189.

\begin{verbatim}
<<
new Staff = "ossia" \with {
  \remove "Time_signature_engraver"
  hide Clef
  fontSize = #-3
  \override StaffSymbol.staff-space = #(magstep -3)
  \override StaffSymbol.thickness = #(magstep -3)
} \relative c'' {
  R1*3
  c4 e8 d c2
}
\new Staff \relative c' {
  c4 b c2
  e4 f e2
  g4 a g2 \break
  c4 b c2
  g4 a g2
  e4 d c2
}
>>
\end{verbatim}
Selected Snippets

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.

\begin{verbatim}
\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 {
        \set 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]
        }
    }>>
\end{verbatim}
See also


Notation Reference: [Hiding staves], page 189.

Snippets: Section “Staff notation” in Snippets.

Internals Reference: Section “StaffSymbol” in Internals Reference.

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 c' <<
  \new Staff {
    e4 f g a \break
    b1 \break
  }

Empty staves can be hidden by setting the \Staff \RemoveEmptyStaves command in the \layout block. In orchestral scores, this style is known as ‘Frenched Score’. By default, this command hides and removes all empty staves in a score except for those in the first system.

\layout {
  \context {
    \Staff \RemoveEmptyStaves
  }
}
\relative c' <<
  \new Staff {
    e4 f g a \break
    b1 \break
}

\layout {
  \context {
    \Staff \RemoveEmptyStaves
  }
}
\relative c' <<
  \new Staff {
    e4 f g a \break
    b1 \break
}

\layout {
  \context {
    \Staff \RemoveEmptyStaves
  }
}
\relative c' <<
  \new Staff {
    e4 f g a \break
    b1 \break
}

\layout {
  \context {
    \Staff \RemoveEmptyStaves
  }
}
\relative c' <<
  \new Staff {
    e4 f g a \break
    b1 \break
}
\Staff \RemoveEmptyStaves can also be used to create ossia sections for a staff. For details, see \Ossia staves, page 185.

The \VaticanaStaff \RemoveEmptyStaves command may be used to hide empty staves in ancient music contexts. Similarly, \RhythmicStaff \RemoveEmptyStaves may be used to hide empty \RhythmicStaff contexts.

Predefined commands
\Staff \RemoveEmptyStaves, \VaticanaStaff \RemoveEmptyStaves, \RhythmicStaff \RemoveEmptyStaves.

Selected Snippets
Removing the first empty line

The first empty staff can also be removed from the score by setting the \VerticalAxisGroup property remove-first. This can be done globally inside the \layout block, or locally inside the specific staff that should be removed. In the latter case, you have to specify the context (\Staff applies only to the current staff) in front of the property.

The lower staff of the second staff group is not removed, because the setting applies only to the specific staff inside of which it is written.

\layout {
  \context {
    \Staff \RemoveEmptyStaves
    % To use the setting globally, uncomment the following line:
  }
}
See also

Music Glossary: Section “Frenched staff” in Music Glossary.
Notation Reference: Section 5.1.5 [Changing context default settings], page 555, [Staff symbol], page 182, [Ossia staves], page 185, [Hidden notes], page 207, [Invisible rests], page 54, Section 5.4.6 [Visibility of objects], page 584.

Snippets: Section “Staff notation” in Snippets.

Internals Reference: Section “ChordNames” in Internals Reference, Section “FiguredBass” in Internals Reference, Section “Lyrics” in Internals Reference, Section “Staff” in Internals Reference, Section “VerticalAxisGroup” in Internals Reference, Section “Staff_symbol_ engraver” in Internals Reference.

Known issues and warnings
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 Section 5.1.5 [Changing context default settings], page 555.

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."
}
{ c4.. g'16 c4.. g'16 \break | c1 }

Violin
\markup can be used to create more complex instrument names:

\new Staff \with {
  instrumentName = \markup {
    \column { "Clarinetti"
      \line { "in B" \smaller \flat }
    }
  }
}
{ c4 c,16 d e f g2 }

\markup
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"
}
{ f2 g4 f }
\new Staff \with {
  instrumentName = \markup {
      \center-column { "Clarinet"
      \line { "in B" \smaller \flat }
    }
  }
}
{ c4 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`, page 501.

```
\relative c' {
  <<
    \new Staff \with {
      instrumentName = #'Alto Flute in G"
      shortInstrumentName = #"Flt."
    }
    { f2 g4 f \break
      g4 f g2 }
    \new Staff \with {
      instrumentName = #"Clarinet"
      shortInstrumentName = #"Clar."
    }
    { c,4 b c2 \break
      c2 b4 c }
  >>
}
```

```
\layout {
```
To add instrument names to other contexts (such as \texttt{ChordNames} or \texttt{FiguredBass}), \texttt{Instrument\_name\_engraver} must be added to that context. For details, see Section 5.1.4 \[Modifying context plug-ins], page 553.

The \texttt{shortInstrumentName} may be changed in the middle of a piece. However, only the first instance of \texttt{instrumentName} will be printed and subsequent changes will be ignored:

\begin{verbatim}
\new Staff \with {
  instrumentName = #'Flute"
  shortInstrumentName = #"Flt."
} { c1 c c c \break c1 c c \break \set Staff.instrumentName = #"Clarinet"
  \set Staff.shortInstrumentName = #"Clt."
  c1 c c \break c1 c c c \break }
\end{verbatim}
If an instrument switch is needed, \texttt{addInstrumentDefinition} may be used in combination with \texttt{instrumentSwitch} to create a detailed list of the necessary changes for the switch. The \texttt{addInstrumentDefinition} command has two arguments: an identifying string, and an association list of context properties and values to be used for the instrument. It must be placed in the toplevel scope. \texttt{instrumentSwitch} is used in the music expression to declare the instrument switch:

\begin{verbatim}
\addInstrumentDefinition \\
"contrabassoon"
#`((instrumentTransposition . ,(ly:make-pitch -1 0 0))
 (shortInstrumentName . "Cbsn.")
(clefGlyph . "clefs.F")
(middleCPosition . 6)
(clefPosition . 2)
(instrumentCueName . ,(make-bold-markup "cbsn."))
(midiInstrument . "bassoon")
\end{verbatim}

\begin{verbatim}
\new Staff \with \\
instrumentName = "Bassoon"
\}
\relative c' \\
  \clef tenor
  \compressFullBarRests
  c2 g'
  R1*16
  \instrumentSwitch "contrabassoon"
  c,,,2 g \break
  c,1 ~ | c1
\}
\end{verbatim}

See also

- Notation Reference: \texttt{\textbackslash paper variables for shifts and indents}, page 501, Section 5.1.4 [Modifying context plug-ins], page 553.
- Snippets: Section “Staff notation” in \texttt{Snippets}.
- Internals Reference: Section “InstrumentName” in \texttt{Internals Reference}, Section “PianoStaff” in \texttt{Internals Reference}, Section “Staff” in \texttt{Internals Reference}.

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 \textit{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.

\begin{verbatim}
fluteNotes = \relative c'' {
    a4 gis g gis | b4"quoted" r8 ais\p a4( f)
}

oboeNotes = \relative c'' {
    c4 cis c b \quoteDuring #"flute" { s1 }
}
\addQuote "flute" { \fluteNotes }
\score {
    <<
        \new Staff \with { instrumentName = "Flute" } \fluteNotes
        \new Staff \with { instrumentName = "Oboe" } \oboeNotes
    >>
}
\end{verbatim}

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.

\begin{verbatim}
fluteNotes = \relative c'' {
    a4 gis g gis | b4"quoted" r8 ais\p a4( f)
}

oboeNotes = \relative c'' {
    c4 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
    >>
}
\end{verbatim}
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.

```latex
clarinetNotes = \relative c'' {
    \transposition bes
    \key d \major
    b4 ais a ais | cis4"quoted" r8 bis\p b4( f)
}

oboeNotes = \relative c'' {
    c4 cis c b \quoteDuring #"clarinet" { s1 }
}
```

```
\addQuote "clarinet" { \clarinetNotes }
```

```
\score {
    \new Staff \with { instrumentName = "Clarinet" } \clarinetNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
}
```

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.

```latex
fluteNotes = \relative c'' {
    a2 g2 |
    b4\"quoted" r8 ais a4\f( c->)
}
```

```latex
oboeNotes = \relative c'' {
    c2. b4 |
    \quoteDuring #"flute" { s1 }
}
```
\addQuote "flute" { \fluteNotes }

\score { 
  \begin{music}
  \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
  \end{music}
}

Quotes can also be tagged, see [Using tags], page 475.

See also

Notation Reference: [Instrument transpositions], page 24, [Using tags], page 475.
Installed Files: ‘scm/define-event-classes.scm’.
Snippets: Section “Staff notation” in Snippets.
Internals Reference: Section “Music classes” in Internals Reference, Section “QuoteMusic” in Internals Reference, Section “Voice” in Internals Reference.

Known issues and warnings

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.

\begin{music}
  \{ e2\rest r4. e8 \}
  \new CueVoice {
    \stemUp d'8""flute" c d e fis2
  }
\end{music}
Chapter 1: Musical notation

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.

\clef "bass"
R1
<<
{ e2\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.

\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 forms of \quoteDuring, see [Quoting other voices], page 195 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 \quotename 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 c'' {
   r2. c4 | d8 c d e fis2 | g2 d |
}

\oboeNotes = \relative c'' {
   R1
   \new \CueVoice { \set \instrumentCueName = "flute" }
   \cueDuring \#"flute" \#UP { R1 }
}
g2 c,
}
\addQuote "flute" { \fluteNotes }
\new Staff {
  \oebNotes
}

\addQuote "flute" { \fluteNotes }
\new Staff {
  \oebNotes
}

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.

Note: 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.

\oebNotes = \relative c'' {
  r2 r8 d16(f f e g f a)
  g8 g16 g g2.
}
\addQuote "oboe" { \oebNotes }

\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,
}

The name of the instrument playing the cue can be printed by setting the instrumentCueName property in a temporary CueVoice context. The placement and style of the instrumentCueName is controlled by the InstrumentSwitch object, see Instrument names, page 192. 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 c'' {
  r2. c4 d8 c d e fis2 g2 d2 
}
\bassoonNotes = \relative c {
\clef bass
\new CueVoice { \set instrumentCueName = "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.

\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.
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], page 475.

fluteNotes = \relative c' {  
  r2. c4 d8 c d e fis2 g2 d2  }

bassoonNotes = \relative c  {
  \clef bass  
  R1  
  \tag #'part {  
    \clef treble  
    \new CueVoice { \set instrumentCueName = "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], page 192.

See also

Notation Reference: [Quoting other voices], page 195, [Instrument transpositions], page 24, [Instrument names], page 192, [Clef], page 16, [Musical cues], page 284, [Using tags], page 475.

Snippets: Section “Staff notation” in Snippets.

Internals Reference: Section “CueVoice” in Internals Reference, Section “Voice” in Internals Reference.

Known issues and warnings
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

The font size of notation elements may be altered. It does not change the size of variable symbols, such as beams or slurs.

| Note: For font sizes of text, see [Selecting font and font size], page 224. |
|\huge c4.\rightarrow d8\rightarrow3 |
|\large c4.\rightarrow d8\rightarrow3 |
|\normalsize c4.\rightarrow d8\rightarrow3 |
|\small c4.\rightarrow d8\rightarrow3 |
|\tiny c4.\rightarrow d8\rightarrow3 |
|\teeny c4.\rightarrow d8\rightarrow3 |

Internally, this sets the fontSize property. This in turn causes the font-size property to be set in all layout objects. The value of font-size is a number indicating the size relative to the standard size for the current staff height. Each step up is an increase of approximately 12% of the font size. Six steps is exactly a factor of two. The Scheme function \magstep converts a font-size number to a scaling factor. The font-size property can also be set directly, so that only certain layout objects are affected.

\set fontSize = #3
c4.\rightarrow d8\rightarrow3
\override NoteHead.font-size = #\-4
c4.→ d8--3
\override Script.font-size = #2
c4.→ d8--3
\override Stem.font-size = #-5
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, a 10pt font is selected.

The font-size property can only be set on layout objects that use fonts. These are the ones supporting the font-interface layout interface.

Predefined commands
\teeny, \tiny, \small, \normalsize, \large, \huge.

See also
Snippets: Section “Editorial annotations” in Snippets.
Internals Reference: Section “font-interface” in Internals Reference.

Fingering instructions
Fingering instructions can be entered using ‘note-digit’:
c4-1 d-2 f-4 e-3

Markup texts or strings may be used for finger changes.
c4-1 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.
<a_	humb a'-3>2 <b_	humb b'-3>

Fingerings for chords can also be added to individual notes by adding them after the pitches.
<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 Section 5.4.2 [Direction and placement], page 577.

Selected Snippets

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>
}

\relative c' {
  <c-1 e-2 g-3 b-5>2
  <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

% Corrected to avoid collisions
r8
\override Fingering.add-stem-support = ##t
<f c'-5>8
\override StringNumber.add-stem-support = ##t
<f c'\5>8
\override StrokeFinger.add-stem-support = ##t
<f c'-\rightHandFinger #2 >8
}

See also

Notation Reference: Section 5.4.2 [Direction and placement], page 577.
Snippets: Section “Editorial annotations” in Snippets.

Hidden notes

Hidden (or invisible or transparent) notes can be useful in preparing theory or composition exercises.
c4 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.

\hideNotes
e8(\p f g a)--
\unhideNotes
e8(\p f g a)--

Predefined commands
\hideNotes, \unHideNotes.

See also
Notation Reference: [Invisible rests], page 54, Section 5.4.6 [Visibility of objects], page 584, [Hiding staves], page 189.
Snippets: Section “Editorial annotations” in Snippets.
Internals Reference: Section “Note_spacing_ engraver” in Internals Reference, Section “NoteSpacing” in Internals Reference.

Coloring objects
Individual objects may be assigned colors. Valid color names are listed in the Section A.7 [List of colors], page 622.
\override NoteHead.color = #red
c4 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.
\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")
\override Accidental.color = #(x11-color 'DarkRed)
gis a
\override NoteHead.color = #(x11-color "LimeGreen")
gis a
% this is deliberate nonsense; note that the stems remain black
\override Stem.color = #(x11-color 'Boggle)
b2 cis

\override Staff.StaffSymbol.color = #(x11-color 'SlateBlue2)
\set Staff.instrumentName = \markup {
  \with-color #(x11-color 'navy) "Clarinet"
}
\override Stem.color = #(rgb-color 0 0 0)
gis8 a
\override Stem.color = #(rgb-color 1 1 1)
gis8 a
\override Stem.color = #(rgb-color 0 0 0.5)
gis4 a

See also

Notation Reference: Section A.7 [List of colors], page 622, Section 5.3.4 [The tweak command], page 571.

Snippets: Section “Editorial annotations” in Snippets.

Known issues and warnings
An X11 color is not necessarily exactly the same shade as a similarly named normal color.

Not all X11 colors are distinguishable in a web browser, i.e., a web browser might not display a difference between LimeGreen and ForestGreen. For web use normal colors are recommended (i.e., blue, green, red).

Notes in a chord cannot be colored with \override; use \tweak instead, see Section 5.3.4 [The tweak command], page 571.

Parentheses
Objects may be parenthesized by prefixing \parenthesize to the music event. When prefixed to a chord, it parenthesizes every note. Individual notes inside a chord may also be parenthesized.
c2 \parenthesize d
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.

c2-\parenthesize -. d
c2 \parenthesize r

See also

Snippets: Section “Editorial annotations” in Snippets.

Internals Reference: Section “Parenthesis engraver” in Internals Reference, Section “ParenthesesItem” in Internals Reference, Section “parentheses-interface” in Internals Reference.

Known issues and warnings

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 Section 5.4.2 [Direction and placement], page 577.

Predefined commands

\stemUp, \stemDown, \stemNeutral.

Selected Snippets

\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.
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.

\[ \text{new Voice with \{ consists "Balloon\_engraver" } \]

\{ 
\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, \texttt{balloonGrobText} and \texttt{balloonText}; the former is used like \texttt{\once \override} to attach text to any grob, and the latter is used like \texttt{\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" } \\
{ \balloonGrobText #'Stem #'(3 . 4) \markup { "I'm a Stem" } \\
a8 \\
\balloonGrobText #'Rest #'(-4 . -4) \markup { "I'm a rest" } \\
r \\
\balloonLengthOn \\
<, g'\-\balloonText #'(-2 . -2) \markup { "I'm a note head" } c>2. \\
}

\begin{center}
\includegraphics[width=0.5\textwidth]{example.png}
\end{center}

Predefined commands
\balloonLengthOn, \balloonLengthOff.

See also
Snippets: Section “Editorial annotations” in Snippets.

Internals Reference: Section “Balloon\_engraver” in Internals Reference, Section “Balloon\-TextItem” in Internals Reference, Section “balloon\-interface” in Internals Reference.

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.

\begin{verbatim}
\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 c' { 
 \stemUp 
 c4. d8 e8 f g4 
 } 
 \new Staff \relative c { 
\end{verbatim}

Selected Snippets

*Grid lines: changing their appearance*

The appearance of grid lines can be changed by overriding some of their properties.

\begin{verbatim}
\score {
\new ChoirStaff <<
\new Staff {
\relative c'' {
\stemUp
\c4 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)
}
\end{verbatim}
Analysis brackets
Brackets are used in musical analysis to indicate structure in musical pieces. Simple horizontal brackets are supported.

```
\layout {
  \context { 
    \Score 
    \consists "Grid_line_span_engraver"
    \override NoteColumn.X-offset = #-0.5
  }
}
```

Analysis brackets may be nested.

```
\layout {
  \context { 
    \Voice 
    \consists "Horizontal_bracket_engraver"
  }
  \relative c'' { 
    c2\startGroup 
    d\stopGroup 
  }
}
```

Analysis brackets may be nested.
\relative c'' {  
c4\startGroup\startGroup  
d4\stopGroup  
e4\startGroup  
d4\stopGroup\stopGroup  
}  

See also  
Snippets: Section “Editorial annotations” in Snippets.  
Internals Reference: Section “Horizontal_bracket_engraver” in Internals Reference, Section “HorizontalBracket” in Internals Reference, Section “horizontal-bracket-interface” in Internals Reference, Section “Staff” in Internals Reference.

1.8 Text
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: Section 2.1 [Vocal music], page 239, Section 3.2 [Titles and headers], page 448.

1.8.1 Writing text
This section introduces different ways of adding text to a score.

**Note:** 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], page 478.

**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 Section 5.4.2 [Direction and placement], page 577.

```latex
a8\"pizz.\" g f e a4\"scherz.\" f
```

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 Section 1.8.2 [Formatting text], page 223.

```latex
a8\markup { \italic pizz. } g f e
a4\markup { \tiny scherz. \bold molto } f
```

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.

```latex
a8\"pizz.\" g f e
\textLengthOn
a4\"scherzando\" f
```

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: Section 2.1 [Vocal music], page 239, Section 3.2 [Titles and headers], page 448.
In addition to text scripts, articulations can be attached to notes. For more information, see [Articulations and ornamentations], page 111.

For more information about the relative ordering of text scripts and articulations, see Section “Placement of objects” in Learning Manual.

Predefined commands
\textLengthOn, \textLengthOff.

See also
Notation Reference: Section 1.8.2 [Formatting text], page 223, Section 5.4.2 [Direction and placement], page 577, [Articulations and ornamentations], page 111.
Snippets: Section “Text” in Snippets.
Internals Reference: Section “TextScript” in Internals Reference.

Known issues and warnings
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:
\override TextSpanner.bound-details.left.text = "rit."
b1\startTextSpan c\stopTextSpan
e\startTextSpan

\text\rit-
\line\rit-
\right
\line\rit-
\right
\line

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 Section 1.8.2 [Formatting text], page 223.
\override TextSpanner.bound-details.left.text = \markup { \upright "rit." }b1\startTextSpan c\stopTextSpan
e\startTextSpan

\text\rit-
\line\rit-
\right
\line\rit-
\right
\line\rit-
\right
\line

The line style, as well as the text string, can be defined as an object property. This syntax is described in Section 5.4.7 [Line styles], page 590.

Predefined commands
\textSpannerUp, \textSpannerDown, \textSpannerNeutral.
Known issues and warnings

LilyPond is only able to handle one text spanner per voice.

Selected Snippets

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 (parser location mymarkup) (markup?)
 (make-music 'CrescendoEvent
 'span-direction START
 'span-type 'text
 'span-text mymarkup))

mydecresc =
#(define-music-function (parser location 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-\mydecresc "custom decresc" c4 c4 c4 |
  c4 c4\! c4 c4}
Text marks

Various text elements may be added to a score using the syntax described in [Rehearsal marks], page 101:

\mark "Allegro"
\c4
\c c c

\begin{center}
Allegro
\end{center}

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 Section 1.8.2 [Formatting text], page 223:

\begin{verbatim}
\begin{music}
\vel \perc
\c2 \\mark \\markup { \italic { colla parte } }
\d2 \e2 <\e2 \g2>
\c2 \f2 <\f2 \aes1>
\end{music}
\end{verbatim}

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], page 233:

\begin{verbatim}
\begin{music}
\vel \perc
\c2 \\mark \\markup { \musicglyph #"scripts.ufermata" }
\e2 \f2 <\aes d>
\end{music}
\end{verbatim}

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.
\mark "Allegro"
c1 c
\mark "assai" \break
c c

Allegro

\markLengthOn, \markLengthOff.

Selected Snippets

*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 "||"
}

D.C. al Fine

*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' }
>>
See also

Notation Reference: [Rehearsal marks], page 101, Section 1.8.2 [Formatting text], page 223, [Music notation inside markup], page 233, Section A.8 [The Feta font], page 624.

Snippets: Section “Text” in Snippets.

Internals Reference: Section “MarkEvent” in Internals Reference, Section “Mark_engraver” in Internals Reference, Section “RehearsalMark” in Internals Reference.

Separate text

A \markup block can exist by itself, outside of any \score block, as a “top-level expression”. This syntax is described in Section 3.1.5 [File structure], page 446.

\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 Section 3.1.2 [Multiple scores in a book], page 443.

\score {
  c'1
}
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
\score {
  c'1
}
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], page 235.

**Predefined commands**

\markup, \markuplist.

**Selected Snippets**

*Stand-alone two-column markup*

Stand-alone text may be arranged in several columns using \markup commands:

\markup {
  \fill-line { 
    \hspace #1 
    \column {
      \line { O 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 { O 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.
1.8.2 Formatting text

This section presents basic and advanced text formatting, using the \( \texttt{\textbackslash markup} \) mode specific syntax.

**Text markup introduction**

A \( \texttt{\textbackslash markup} \) block is used to typeset text with an extensible syntax called “markup mode”.

The markup syntax is similar to LilyPond’s usual syntax: a \( \texttt{\textbackslash markup} \) 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, \( \texttt{\textbackslash markup} \) blocks may contain nested expressions or markup commands, entered using the backslash \( \texttt{\textbackslash} \) character. Such commands only affect the first following expression.

\[
a1^-\texttt{\textbackslash markup intenso}
\]
\[
a2^-\texttt{\textbackslash markup \{ poco \italic più forte \}}
\]
\[
c e1
\]
\[
d2^-\texttt{\textbackslash markup \{ \italic "string. assai" \}}
\]
\[
e
\]
\[
b1^-\texttt{\textbackslash markup \{ \bold \{ molto \italic agitato \} \}}
\]
\[
c
\]

\[
c
\]

\[
\texttt{\textcolor{red}{\textbackslash italic markup...}}
\]
\[
a \texttt{\textbackslash\{ \italic "... prints \"italic\" letters!" \}}
\]
\[
a a
\]

\[
\texttt{\textcolor{red}{\textbackslash italic markup...}}
\]
\[
... \texttt{\textcolor{red}{prints "italic" letters!}}
\]

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 \( \texttt{\textbackslash markup} \) expression is treated the same as the first one:

\[
c1^-\texttt{\textbackslash markup \{ \center-column \{ a bbb c \} \}}
\]
\[
c1^-\texttt{\textbackslash markup \{ \center-column \{ a \{ bbb c \} \} \}}
\]
\[
c1^-\texttt{\textbackslash markup \{ \center-column \{ a \textbackslash\{line \{ bbb c \} \} \} \}}
\]
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 } 
```

![Allegro](image)

An exhaustive list of \markup-specific commands can be found in Section A.10 [Text markup commands], page 645.

### See also

- Notation Reference: Section A.10 [Text markup commands], page 645.
- Snippets: Section “Text” in Snippets.
- Installed Files: ‘scm/markup.scm’.

### Known issues and warnings

Syntax errors for markup mode can be confusing.

### Selecting font and font size

Basic font switching is supported in markup mode:

```d1`\markup \{ \bold \{ Più mosso \} \italic \{ non troppo \underline Vivo \} \} 
\{ r2 r4 r8 \}
```

![Più mosso non troppo Vivo](image)

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,
It can be set relative to its previous value,

\markup { 
\larger Sinfonia 
\smaller da 
\magnify #0.6 camera 
}

It can be increased or decreased relative to the value set by the global staff size,

\markup { 
\fontsize #-2 Sinfonia 
\fontsize #1 da 
\fontsize #3 camera 
}

It can also be set to a fixed point-size, regardless of the global staff size,

\markup { 
\abs-fontsize #20 Sinfonia 
\abs-fontsize #8 da 
\abs-fontsize #14 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) } 
}
}
```

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.

```
1^{st} movement
1^{st} movement (part two)
```
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], page 119 and [Manual repeat marks], page 145.

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:

1\textsuperscript{st} movement
\textit{p, con dolce espressione}

An exhaustive list of font switching commands and custom font usage commands can be found in Section A.10.1 [Font], page 645.

Defining custom font sets is also possible, as explained in Section 1.8.3 [Fonts], page 236.

Predefined commands
\texttt{teeny, tiny, small, normalsize, large, huge, smaller, larger}.

See also

Notation Reference: Section A.10.1 [Font], page 645, [New dynamic marks], page 119, [Manual repeat marks], page 145, Section 1.8.3 [Fonts], page 236.

Installed Files: `scm/define-markup-commands.scm`

Snippets: Section “Text” in Snippets.

Internals Reference: Section “TextScript” in Internals Reference.
Known issues and warnings

Using the font sizing commands \texttt{\textsc{teeny}}, \texttt{\textsc{tiny}}, \texttt{\textsc{small}}, \texttt{\textsc{normalsize}}, \texttt{\textsc{large}}, and \texttt{\textsc{huge}} will lead to inconsistent line spacing compared to using \texttt{\textsc{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 Section “Moving objects” in Learning Manual.

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.

\begin{verbatim}
d1-\markup { poco }
f
d-\markup { \left-align poco }
f
d-\markup { \center-align { poco } }
f
d-\markup { \right-align poco }
\end{verbatim}

\begin{verbatim}
\begin{tikzpicture}
\node at (0,0) {\texttt{\textsc{poco}}};
\node at (1,0) {\texttt{\textsc{poco}}};
\node at (2,0) {\texttt{\textsc{poco}}};
\node at (3,0) {\texttt{\textsc{poco}}};
\end{tikzpicture}
\end{verbatim}

Horizontal alignment may be fine-tuned using a numeric value:

\begin{verbatim}
a1-\markup { \halign #-1 poco }
ea'
a,-\markup { \halign #0 poco }
ea'
a,-\markup { \halign #0.5 poco }
ea'
a,-\markup { \halign #2 poco }
\end{verbatim}

\begin{verbatim}
\begin{tikzpicture}
\node at (0,0) {\texttt{\textsc{poco}}};
\node at (1,0) {\texttt{\textsc{poco}}};
\node at (2,0) {\texttt{\textsc{poco}}};
\node at (3,0) {\texttt{\textsc{poco}}};
\end{tikzpicture}
\end{verbatim}

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], page 219.

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.

\begin{verbatim}
d2^-\markup {
  \texttt{Acte I}
  \raise #2 { Scène 1 }
}
a'
g_\markdown {
  \null
}\end{verbatim}
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:

```latex
\lower #4 \bold { Très modéré }
\markup { a d, \raise #4 \italic { Une forêt. } }
\general-align #Y #3.2 \bold "Très modéré"
\translate-scaled #'(-1 . 2) \teeny "Une forêt."
```

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:

```latex
\markup {
  \column {
    a "b c"
    \line { d e f }
  }
  \hspace #10
  \center-column {
    a "b c"
    \line { 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:

```latex
\markup {
  \fill-line {
    \line { William S. Gilbert }
    \center-column {
      \huge \smallCaps "The Mikado"
      or
      \smallCaps "The Town of Titipu"
    }
    \line { Sir Arthur Sullivan }
  }
}
```

William S. Gilbert \textit{The Mikado} Sir Arthur Sullivan
\textit{The Town of Titipu}

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.

```latex
\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 Section A.10.2 [Align], page 654.

See also


Notation Reference: Section A.10.2 [Align], page 654, [Text marks], page 219.

Installed Files: ‘scm/define-markup-commands.scm’.

Snippets: Section “Text” in Snippets.

Internals Reference: Section “TextScript” in Internals Reference.

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 \text \bold Prelude
}
}
Chapter 1: Musical notation

Jack
in the box

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 Section A.10.2 [Align], page 654.

\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 {
}
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:

```plaintext
c1^\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"  
}
```

An exhaustive list of graphics-specific commands can be found in Section A.10.3 [Graphic], page 669.

See also

Notation Reference: Section A.10.3 [Graphic], page 669, Section 1.7 [Editorial annotations], page 204, Section A.10.2 [Align], page 654.


Snippets: Section “Text” in Snippets.

Internals Reference: Section “TextScript” in Internals Reference.
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:

```latex
a2 \markup \note \#4 \#1 = \note-by-number \#1 \#1 \#1.5
b1 \markup \natural \semiflat \flat \sesquiflat \doubleflat
\glissando
a1 \markup \natural \semisharp \sharp \sesquisharp \doublesharp
\glissando b
```

Other notation objects may also be printed in markup mode:

```latex
g1 bes
ee\markup \tie-lyric \#4~1
fis \markup \dynamic rf
bes \markup \beam \#8 \#0.1 \#0.5
\markalphabet \#8
\markletter \#8
\timesig.C44\clefs.G
```

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 Section A.8 [The Feta font], page 624.

```
c2
c' \markup \musicglyph \#eight
\musicglyph \clefs.G_change
```
Another way of printing non-text glyphs is described in [Fonts explained], page 236. This is useful for printing braces of various sizes.

The markup mode also supports diagrams for specific instruments:

Such diagrams are documented in Section A.10.5 [Instrument Specific Markup], page 682.

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:
An exhaustive list of music notation related commands can be found in Section A.10.4 [Music], page 676.

See also

Notation Reference: Section A.10.4 [Music], page 676, Section A.8 [The Feta font], page 624, [Fonts explained], page 236.

Installed Files: `scm/define-markup-commands.scm`, `scm/fret-diagrams.scm`, `scm/harp-pedals.scm`.

Snippets: Section “Text” in Snippets.

Internals Reference: Section “TextScript” in Internals Reference.

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 Section A.11 [Text markup list commands], page 696.

See also

Notation Reference: Section A.11 [Text markup list commands], page 696.
Extending LilyPond: Section “New markup list command definition” in Extending.

Installed Files: `scm/define-markup-commands.scm`.

Snippets: Section “Text” in Snippets.

Internals Reference: Section “TextScript” in Internals Reference.

Predefined commands

\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:

```plaintext
a1^\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], page 233.

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 Section 5.4.8 [Rotating objects], page 590.

Three families of text fonts are made available: the roman (serif) font, that defaults to New Century Schoolbook, the sans font and the monospaced typewriter font – these last two families are determined by the Pango installation.

Note: There are no default fonts associated with the sans and typewriter font-families. An input file that specifies either of these can lead to different output on different computers. To ensure consistent output among multiple platforms, fonts must be specified by name, and those fonts must be available on any system that processes the file. See [Single entry fonts], page 237 and [Entire document fonts], page 238.

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.

```plaintext
\override Score.RehearsalMark.font-family = #'typewriter
\mark \markup "Ouverture"
\override Voice.TextScript.font-shape = #'italic
```
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], page 224:

\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], page 237 and [Entire document fonts], page 238.

**See also**

Notation Reference: Section A.8 [The Feta font], page 624, [Music notation inside markup], page 233, Section 5.4.8 [Rotating objects], page 590, [Selecting font and font size], page 224, Section A.10.1 [Font], page 645.

**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
The following command displays a list of all available fonts on the operating system:
```
lilypond -dshow-available-fonts x
```

See also
Notation Reference: [Fonts explained], page 236, [Entire document fonts], page 238.
Snippets: Section “Text” in Snippets.

Entire document fonts
It is possible to change the fonts to be used as the default fonts in the *roman*, *sans* and *typewriter* 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. For an explanation of fonts, see [Fonts explained], page 236.
```
\paper { 
  #(define fonts
    (make-pango-font-tree "Times New Roman"
      "Nimbus Sans"
      "Luxi Mono"
      (/ staff-height pt 20)))
  
\relative c'{
  c1-\markup { roman,
    \sans sans,
    \typewriter typewriter. }
}
```

See also
Notation Reference: [Fonts explained], page 236, [Single entry fonts], page 237, [Selecting font and font size], page 224, Section A.10.1 [Font], page 645.
2 Specialist notation

This chapter explains how to create musical notation for specific types of instrument or in specific styles.

2.1 Vocal music

This section explains how to typeset vocal music, and make sure that the lyrics will be aligned with the notes of their melody.

2.1.1 Common notation for vocal music

This section discusses issues common to most types of vocal music.

References for vocal music

This section indicates where to find details of notation issues that may arise in any type of vocal music.

- Most styles of vocal music use written text as lyrics. An introduction to this notation is to be found in Section “Setting simple songs” in Learning Manual.
- Vocal music is likely to require the use of markup mode, either for lyrics or for other text elements (characters’ names, etc.) This syntax is described in [Text markup introduction], page 223.
- Ambitus may be added at the beginning of vocal staves, as explained in [Ambitus], page 32.
- 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, as explained in [Score layouts for choral], page 279.
See also

Notation Reference: [Text markup introduction], page 223, [Ambitus], page 32, [Score layouts for choral], page 279.
Snippets: Section “Vocal music” in Snippets.

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], page 245. The latter method is described in [Automatic syllable durations], page 243.

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 Section 3.3.3 [Special characters], page 478.

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 c' { \time 3/4 e4 e4. e8 d4 e d c2. } \addlyrics { ","I am so lone -- "$\,$" 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 Section 1.8.2 [Formatting text], page 223.

Selected Snippets

**Formatting lyrics syllables**

Markup mode may be used to format individual syllables in lyrics.

```latex
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
>>

Lyrics *can* contain **Markup**!

See also

- Notation Reference: [Automatic syllable durations], page 243, Section 1.8.3 [Fonts], page 236, Section 1.8.2 [Formatting text], page 223, Section 5.1.1 [Contexts explained], page 545, Section 5.4.1 [Input modes], page 575, [Manual syllable durations], page 245, Section 3.3.3 [Special characters], page 478.
- Internals Reference: Section “LyricText” in Internals Reference.
- Snippets: Section “Text” in Snippets.

**Aligning lyrics to a melody**

Lyrics are printed by interpreting them in the context called Lyrics, see Section 5.1.1 [Contexts explained], page 545.

```latex
\new Lyrics \lyricmode { ... }
```

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], page 243.

<<
\new Staff <<
\time 2/4
\new Voice = "one" \relative c'' {
\voiceOne
c4 b8. a16 g4. r8 a4 ( b ) c2
}
\new Voice = "two" \relative c' {
\voiceTwo
s2 s4. f8 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.
}

\new Voice = "one" \relative c'' {
\time 2/4
c4 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 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 Section 2.1.3 [Stanzas], page 270.

- Lyrics can be aligned independently of the duration of any notes if the durations of the syllables are specified explicitly, and entered with \lyricmode.
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], page 245. This technique is also useful when setting dialogue over music; for examples showing this, see [Dialogue over music], page 288.

When entered in this way the words are left-aligned to the notes by default, but may be center-aligned to the notes of a melody by specifying an associated voice, if one exists. For details, see [Manual syllable durations], page 245.

See also


Notation Reference: Section 5.1.1 [Contexts explained], page 545, [Automatic syllable durations], page 243, Section 2.1.3 [Stanzas], page 270, [Manual syllable durations], page 245, [Dialogue over music], page 288, [Manual syllable durations], page 245.

Internals Reference: Section “Lyrics” in Internals Reference.

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], page 251.

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 Section 5.1.3 [Keeping contexts alive], page 550.

Using \lyricsto

Lyrics can be aligned under a melody automatically by specifying the named Voice context containing the melody with \lyricsto:

```latex
<<
\new Voice = "melody" {
   a1 a4. a8 a2
}
\new Lyrics \lyricsto "melody" {
   These are the words
} >>
```

Joy to the earth!
Life is love, live life.
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, so the \lyricmode keyword may be omitted. By default, the lyrics are placed underneath the notes. For other placements, see [Placing lyrics vertically], page 253.

**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' { c2 e4 g2. } 
  \addlyrics { play the game } 
}
```

More stanzas can be added by adding more \addlyrics sections:

```
{ 
  \time 3/4 
  \relative c' { c2 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 c' { \voiceOne c4 b8. a16 g4. r8 a4 ( b ) c2 }
\new Voice = "two" \relative c' { \voiceTwo s2 s4. f8 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.
}
>>

No more let sins and sorrows grow.

See also

Notation Reference: [Extenders and hyphens], page 251, Section 5.1.3 [Keeping contexts alive], page 550, [Placing lyrics vertically], page 253.

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.

By default, syllables will be left-aligned to the corresponding musical moment. 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:
This technique is useful when writing dialogue over music, see [Dialogue over music], page 288.

To center-align syllables on the notes at the corresponding musical moments, set `associatedVoice` to the name of the Voice context containing those notes. When `associatedVoice` is set, both double hyphens and double underscores can be used to draw hyphenated lines and extenders under melismata correctly.
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.

\new Lyrics \lyricmode {
  \set associatedVoice = #'"melody"
  play the game.
}

See also

Notation Reference: [Dialogue over music], page 288.
Internals Reference: Section “Lyrics” in Internals Reference, Section “Voice” in Internals Reference.

Multiple notes to one syllable

Sometimes, particularly in Medieval and baroque music, several notes are sung on one syllable; this is called melisma, see Section “melisma” in Music Glossary. The syllable to a melisma is usually left-aligned with the first note of the melisma.

See also

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" {
  \time 3/4
  f4 g2 ~ |
  g4 e2 ~ |
  e8
  }
  \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" {
  \time 3/4
  f4 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], page 78.

  \new Voice = "melody" {
  \time 3/4
  \autoBeamOff
  f4 g8[ f e f]
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" {
  \time 3/4
  f4 g8
  \melisma
  f e f
  \melismaEnd
  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`:
Chapter 2: Specialist notation

<<
\new Voice = "melody" {
  \time 3/4
  \set melismaBusyProperties = #'()
  c4 d ( e )
  g8 [ f ] f4 ~ f
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- e e -- le -- i -- son
}
>>

Ky ri e e le i son

Other settings for \texttt{melismaBusyProperties} can be used to selectively include or exclude ties, slurs, and beams from the automatic detection of melismata; see \texttt{melismaBusyProperties} in Section “Tunable context properties” in \textit{Internals Reference}.

Alternatively, if all melismata indications are to be ignored, \texttt{ignoreMelismata} may be set true; see [Stanzas with different rhythms], page 271.

If a melisma is required during a passage in which \texttt{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" {
  \time 3/4
  \set melismaBusyProperties = #'()
  c4 d ( e )
  g8 [ f ] ~ f4 ~ f
}
\new Lyrics \lyricsto "melody" {
  Ky -- ri -- _ e _ _ _
}
>>

Ky ri _ e _ _ _

Predefined commands
\texttt{\autoBeamOff}, \texttt{\autoBeamOn}, \texttt{\melisma}, \texttt{\melismaEnd}.

See also

Musical Glossary: Section “melisma” in \textit{Music Glossary}.
Notation Reference: [Aligning lyrics to a melody], page 241, [Automatic syllable durations], page 243, [Setting automatic beam behavior], page 78, [Stanzas with different rhythms], page 271.
Internals Reference: Section “Tunable context properties” in Internals Reference.

Known issues and warnings
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).

Note: 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.

See also
Internals Reference: Section “LyricExtender” in Internals Reference, Section “LyricHyphen” in Internals Reference.

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:

\new Voice = \relative c'' { 
  c4 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:

\begin{verbatim}
sopranoMusic = \relative c'' { c4 c c c }  
contraltoMusic = \relative c'' { a4 a a a }  
sopranoWords = \lyricmode { Sop -- ra -- no words }  
contraltoWords = \lyricmode { Con -- tral -- to words }  
\end{verbatim}

\begin{verbatim}
\score {  
\new ChoirStaff <<  
\new Staff {  
\new Voice = "sopranos" {  
\sopranoMusic  
}  
}  
\new Lyrics = "sopranos"  
\new Staff {  
\new Voice = "contraltos" {  
\contraltoMusic  
}  
}  
\context Lyrics = "sopranos" {  
\lyricsto "sopranos" {  
\sopranoWords  
}  
}  
\context Lyrics = "contraltos" {  
\lyricsto "contraltos" {  
\contraltoWords  
}  
}  
}  
>>  
\end{verbatim}

See also

Notation Reference: [Placing lyrics vertically], page 253.

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 {
    \relative c'' { c4 c c c }
  }
  \new Lyrics {
    \lyricsto "melody" {
      Here are the words
    }
  }
>>
}

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'' { c4 c c c }
    }
  }
  \new Lyrics \with { alignAboveContext = "staff" } {
    \lyricsto "melody" {
      Here are the words
    }
  }
>>
}

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:
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'' { c4 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 c'' { a4 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 Section “Vocal ensembles templates” in Learning Manual.

Selected Snippets

Obtaining 2.12 lyrics spacing in newer versions

The vertical spacing engine changed for 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' {%
    \score % VERSE ONE
    fis4 fis fis | \break
    fis4. e8 e4
}

altoMusic = \relative c' {%
    \score % VERSE ONE
    d4 d d |
    d4. b8 b4 |
}

tenorMusic = \relative c' {%
    \score
    a4 a a |
    b4. g8 g4 |
}

bassMusic = \relative c {%
    \score
d4 d d |
    g,4. g8 g4 |
}

words = \lyricmode {
    Great is Thy faithfulness,
}
\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))
  }
}
See also


Notation Reference: Section 5.1.7 [Context layout order], page 562, Section 5.1.2 [Creating and referencing contexts], page 547.

Placing syllables horizontally

To increase the spacing between lyrics, set the minimum-distance property of LyricSpace.

```
\override Lyrics.LyricSpace.minimum-distance = #1.0
```

```
\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 c  
  }  
  \addlyrics {  
    longtext longtext longtext longtext  
    longtext longtext longtext longtext  
  }  
  \layout {  
    \context {  
      \Lyrics  
      \override LyricSpace.minimum-distance = #1.0  
    }  
  }  
}

Selected Snippets

*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  
}  
\addlyrics {  
  \once \override LyricText.self-alignment-X = #LEFT  
  "This is left-aligned"  
  \once \override LyricText.self-alignment-X = #CENTER  
  "This is centered"  
  \once \override LyricText.self-alignment-X = #1  
  "This is right-aligned"  
}
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 Section 1.4 [Repeats], page 137. 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 c'' {
        a4 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 c'' {
      a4 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. }
  }
}

>>

\[\text{Not repeated. Repeated twice. Repeated twice.}\]

If the repeated section is to be unfolded and has different words, simply enter all the words:
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative c'' {
          a4 a a a
          \repeat unfold 2 { b4 b b b }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Not re--peat--ed.
        The first time words.
        Second time words.
      }
    }
  >>
}

\[\text{Not repeated. The first time words. Second time words.}\]

When the words to a repeated volta section are different, the words to each repeat must be entered in separate \texttt{Lyrics} contexts, correctly nested in parallel sections:
More verses may be added in a similar way:

\score {
  \new Staff {
    \new Voice = "singleVoice" {
      \relative c'' {
        a4 a a a
        \repeat volta 3 { b4 b b b } c4 c c c
      }
    }
  }
}
\new Lyrics \lyricsto "singleVoice" {
  Not repeated.  The first time words.
  \new Lyrics {
    \set associatedVoice = "singleVoice"
    Sec -- ond time words.
  }
  \new Lyrics {
    \set associatedVoice = "singleVoice"
    The third time words.
  }
}
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`:

```latex
\score { << \\
  \new Staff { \\
    \new Voice = "melody" { \\
      \relative c'' { \\
        a4 a a a \\
        \repeat volta 3 { b4 b b b } \\
        c4 c c c \\
      } \\
    } \\
  } \\
  \new Lyrics = "firstVerse" \lyricsto "melody" { \\
    Not repeated. \ The first time words. \ The end section. \\
    \new Lyrics = "secondVerse" << \\
    \with { \set associativedVoice = "melody" } { \\
      Sec -- ond time words. \ \\
    } \\
  } \\
  \new Lyrics = "thirdVerse" << \\
  \with { \set associativedVoice = "melody" } { \\
    The third time words. \\
  } \\
  >> \\
  The end sec -- tion. \\
  \new Voice = "harmony" { \\
    \relative c' { \\
      f4 f f f \repeat volta 2 { g8 g g4 g2 } a4 a8. a16 a2 \\
    } \\
  } \\
  >> \\
} 
```
Chapter 2: Specialist notation

Repeats with alternative endings

If the words of the repeated section are the same, exactly the same structure can be used for both the lyrics and music.

\score {
  \new Staff {
    \time 2/4
    \new Voice = "melody" {
      \relative c'' {
        a4 a a a
      }
      \repeat volta 2 { b4 b }
      \alternative { { b b } { b c } }
    }
  }
  \new Lyrics {
    \lyricsto "melody" {
      \repeat volta 2 { Re-peat--ed. }
      \alternative { { ed twice. } { ed twice. } }
    }
  }
}

But when the repeated section has different words, 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.

Note: 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.
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 c'' { 
        \repeat volta 2 { b4 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. 
    } 
  } 
  >> 
}
Note that if \texttt{unfoldRepeats} is used around a section containing \texttt{repeatTie}, the \texttt{repeatTie} should be removed to avoid both types of tie being printed.

When the repeated section has different words a \texttt{repeat} cannot be used around the lyrics and \texttt{skip} commands need to be inserted manually, as before.

\begin{verbatim}
\score {
  \new Staff {
    \time 2/4
    \new Voice = "melody" {
      \relative c'' {
        \repeat volta 2 { b4 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.
    }
  }
}>>
\end{verbatim}
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 c'' {
        \repeat volta 2 { b4 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.
    }
  }

}
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 _
    }
  >>
}

They shall not o-ver-come
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 c' {
            <<
                \voiceOne
                e4 e8 e
            }
            \new Voice = "splitpart" {
                \voiceTwo
                c4 c
            }
        }
        \oneVoice
        c4 c |
        c
    }
}
Chapter 2: Specialist notation

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], page 159.

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 __ }
The NullVoice context must be placed within a Staff context and contain notes that are already being displayed in that staff and that are also in the same octave. Otherwise the NullVoice may interact with the printed voices in unexpected ways. For example, arbitrary notes in the NullVoice may cause accidentals to appear (or disappear) on the staff.

This method also can be used with the partcombine function, which does not allow lyrics on its own:

\begin{verbatim}
\new Staff <<
  \new Voice \partcombine \soprano \alto
  \new NullVoice = "aligner" \aligner
  \new Lyrics \lyricsto "aligner" \words
>>
\end{verbatim}

Known issues and warnings

The addLyrics function only works with Voice lyrics and so cannot be used with NullVoice.

The partcombine function is described in [Automatic part combining], page 167.

Lastly, this method can be used even when the voices are in different staves, and is not limited to only two voices:

\begin{verbatim}
\new ChoirStaff <<
  \new Staff <<
\end{verbatim}
\new NullVoice = "aligner" \aligner

\new Lyrics \lyricsto "aligner" \words
\new Staff \partcombine \altoOne \altoTwo

However, note that in the second half of the measure above, the notes in the NullVoice context reflect the rhythm of the lower staff, but they do not deviate from the single pitch being displayed in the staff to which the NullVoice belongs. While not actually required in this particular example, it is a good idea in general to enter the notes in this way.

2.1.3 Stanzas

Adding stanza numbers

Stanza numbers can be added by setting \stanza, e.g.,

\new Voice {
  \time 3/4 g2 e4 a2 f4 g2.
} \addlyrics {
  \set stanza = #"1. "
  Hi, my name is Bert.
} \addlyrics {
  \set stanza = #"2. "
  Oh, ché -- ri, je t’aime
}

1. Hi, my name is Bert.
2. Oh, ché - ri, je t’aime

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
  }
}

}
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 {\time 3/4 g2 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 c' \new Voice = "lahlah" {\set Staff.autoBeaming = ##f c4 \slurDotted f8.[( g16)] a4} \new Lyrics \lyricsto "lahlah" {\text}>>
more slow -- ly
}
\new Lyrics \lyricsto "lahlah" {
go
\set ignoreMelismata = ##t
fas -- ter
\unset ignoreMelismata
still
}

\set ignoreMelismata = ##t
\new Voice = melody \relative c' {
f4 \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.
}

Known issues and warnings
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:

```
\set ignoreMelismata = ##t
\new Voice = melody \relative c' {
f4 \appoggiatura a32 b4
\grace { f16 a16 } b2
\afterGrace b2 { f16[ a16] }
\appoggiatura a32 b4
\acciaccatura a8 b4
}

more slow - ly
go fas-ter still

normal case, grace case, after grace case, app. case, acc. case.
Known issues and warnings

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 c' \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
c4
    <<
    \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 --
```
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 \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.

\score{ <<
  \new Voice = "one" { \melody }
  \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
}


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.

\begin{verbatim}
melody = \relative c' {
    c4 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 { }
\end{verbatim}

1. Mary had a little lamb, its fleece was white as snow.
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.
See also

Internals Reference: Section “LyricText” in Internals Reference, Section “StanzaNumber” in Internals Reference.

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 Section 1.6.1 [Displaying staves], page 175.
- For writing piano music, see Section 2.2 [Keyboard and other multi-staff instruments], page 301.
- For writing the lyrics to a melody line, see Section 2.1.1 [Common notation for vocal music], page 239.
- For placing the lyrics, see [Placing lyrics vertically], page 253.
- For entering stanzas, see Section 2.1.3 [Stanzas], page 270.
- Songs are frequently printed with the chording indicated by chord names above the staves. This is described in Section 2.7.2 [Displaying chords], page 389.
- To print fret diagrams of the chords for guitar accompaniment or accompaniment by other fretted instruments, see “Fret diagram markups” in Section 2.4.1 [Common notation for fretted strings], page 316.

See also


Notation Reference: Section 2.1.1 [Common notation for vocal music], page 239, Section 2.7.2 [Displaying chords], page 389, Section 1.6.1 [Displaying staves], page 175, Section 2.2 [Keyboard and other multi-staff instruments], page 301, [Placing lyrics vertically], page 253, Section 2.1.3 [Stanzas], page 270.

Snippets: Section “Vocal music” in Snippets.

Lead sheets

Lead sheets may be printed by combining vocal parts and ‘chord mode’; this syntax is explained in Section 2.7 [Chord notation], page 384.

Selected Snippets

Simple lead sheet

When put together, chord names, a melody, and lyrics form a lead sheet:

```plaintext
<<
\chords { c2 g:sus4 f e }
\relative c''{
   a4 e c8 e r4
   b2 c4( d)
}
```
See also

Notation Reference: Section 2.7 [Chord notation], page 384.

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 Section “Four-part SATB vocal score” in Learning Manual.
- Several templates suitable for various styles of choral music can also be found in the Learning Manual, see Section “Vocal ensembles templates” in Learning Manual.
- For information about ChoirStaff and PianoStaff see [Grouping staves], page 176.
- Shape note heads, as used in Sacred Harp and similar notation, are described in [Shape note heads], page 37.
- 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], page 159.
- When a vocal part temporarily splits, you should use Temporary polyphonic passages (see [Single-staff polyphony], page 159).

**Predefined commands**

\oneVoice, \voiceOne, \voiceTwo.

See also


Notation Reference: Section 5.1.7 [Context layout order], page 562, [Grouping staves], page 176, [Shape note heads], page 37, [Single-staff polyphony], page 159.

Snippets: Section “Vocal music” in Snippets.

Internals Reference: Section “ChoirStaff” in Internals Reference, Section “Lyrics” in Internals Reference, Section “PianoStaff” in Internals Reference.
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 Section 4.2.2 [Setting the staff size], page 506.
- The distances between the systems, the staves and the lyrics can all be adjusted independently. See Section 4.4 [Vertical spacing], page 515.
- 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 Section 4.6 [Fitting music onto fewer pages], page 541.
- 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], page 181.
- For details of other page formatting properties, see Section 4.1 [Page layout], page 494.

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 \texttt{dynamicUp} does this for the dynamic markings in a single \texttt{Voice} context. If there are many \texttt{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:

\[
\texttt{score} \{
  \texttt{\new ChoirStaff <<}
  \texttt{\new Staff \{}
    \texttt{\relative c'' \{ g4\f g g g \}}
  \texttt{\}}
  \texttt{\new Staff \{}
    \texttt{\new Voice \{}
      \texttt{\relative c' \{ d4 d d\p d \}}
    \texttt{\}}
  \texttt{\}}
  \texttt{\layout \{}
    \texttt{\context \{}
      \texttt{\Score}
      \texttt{\override DynamicText.direction = #UP}
      \texttt{\override DynamicLineSpanner.direction = #UP}
    \texttt{\}}
  \texttt{\}}
\]

\[\begin{array}{c}
  f \\
  p \\
\end{array}\]
Predefined commands
\dynamicUp, \dynamicDown, \dynamicNeutral.

See also
Notation Reference: Section 4.6.2 [Changing spacing], page 542, Section 4.6.1 [Displaying spacing], page 541, Section 4.6 [Fitting music onto fewer pages], page 541, Section 4.1 [Page layout], page 494, Section 4.2 [Score layout], page 504, [Separating systems], page 181, Section 4.2.2 [Setting the staff size], page 506, Section 4.3.8 [Using an extra voice for breaks], page 513, Section 4.4 [Vertical spacing], page 515.

Internals Reference: Section “VerticalAxisGroup” in Internals Reference, Section “StaffGrouper” in Internals Reference.

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 ~ 
        a4 \bar "." 
      }
    >>
  }
  \layout { ragged-right = ##t }
}
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], page 176. To nest groups of staves see [Nested staff groups], page 179.
- The printing of empty staves in conductors’ scores and vocal scores is often suppressed. To create such a “Frenched score” see [Hiding staves], page 189.
- Writing orchestral parts is covered in Section 1.6.3 [Writing parts], page 192. Other sections in the Specialist notation chapter may be relevant, depending on the orchestration used. Many instruments are transposing instruments, see [Instrument transpositions], page 24.
- 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], page 181.
- For details of other page formatting properties, see Section 4.1 [Page layout], page 494.
- Dialogue cues, stage directions and footnotes can be inserted, see Section 3.2.3 [Creating footnotes], page 460 and Section 1.8 [Text], page 215. Extensive stage directions can also be added with a section of stand-alone markups between two `score` blocks, see [Separate text], page 221.

See also


Notation Reference: Section 3.2.3 [Creating footnotes], page 460, [Grouping staves], page 176, [Hiding staves], page 189, [Instrument transpositions], page 24, [Nested staff groups], page 179, Section 4.1 [Page layout], page 494, [Separating systems], page 181, [Transpose], page 10, Section 1.6.3 [Writing parts], page 192, Section 1.8.1 [Writing text], page 216.

Snippets: Section “Vocal music” in Snippets.
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 c' { 
      \clef "G_8" 
      c4 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 c' { 
      a4 a a a 
      a4 a a a 
    } 
  } 
} 

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.

\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"
Alternatively, if there are many character changes, it may be easier to set up “instrument”
definitions for each character at the top level so that \instrumentSwitch can be used to indicate
each change.

\addInstrumentDefinition #"kaspar"
#`((instrumentTransposition . ,(ly:make-pitch -1 0 0))
  (shortInstrumentName . "Kas.")
  (clefGlyph . "clefs.G")
  (clefTransposition . -7)
  (middleCPosition . 1)
  (clefPosition . -2)
  (instrumentCueName . ,(markup #:fontsize 1 #:smallCaps "Kaspar"))
  (midiInstrument . "voice oohs"))

\addInstrumentDefinition #"melchior"
#`((instrumentTransposition . ,(ly:make-pitch 0 0 0))
  (shortInstrumentName . "Mel.")
  (clefGlyph . "clefs.F")
  (clefTransposition . 0)
  (middleCPosition . 6)
  (clefPosition . 2)
  (instrumentCueName . ,(markup #:fontsize 1 #:smallCaps "Melchior"))
  (midiInstrument . "choir aahs"))

\relative c' {
  \instrumentSwitch "kaspar"
c4 c c c
  \instrumentSwitch "melchior"
a4 a a a
  \instrumentSwitch "kaspar"
c4 c c c
}

See also

Notation Reference: [Instrument names], page 192, Section A.21 [Scheme functions],
page 747, Section 1.8 [Text], page 215, Section A.10 [Text markup commands], page 645.
Extending LilyPond: Section “Markup construction in Scheme” in Extending.
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], page 195 and [Formatting cue notes], page 198. 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.

```plaintext
flute = \relative c'' {  
s4 s4 e g  
}  
\addQuote "flute" { \flute }

pianoRH = \relative c'' {  
c4. g8  
% position name of cue-ing instrument just before the cue notes,  
% and above the staff  
\new CueVoice {  
  \override InstrumentSwitch.self-alignment-X = #RIGHT  
  \set instrumentCueName = "Flute"  
}  
\cueDuring "flute" #UP { g4 bes4 }  
}
pianoLH = \relative c { c4 <c' e> e, <g c> }
```

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. Note also that the piano right-hand voice is explicitly declared. This is because the
cue notes in this example begin at the start of the first bar and this would otherwise cause the entire piano right-hand notes to be placed in a `CueVoice` context.

```music
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
  \new CueVoice {
    \override InstrumentSwitch.self-alignment-X = #RIGHT
    \override InstrumentSwitch.direction = #DOWN
    \set instrumentCueName = "Clar."
  }
  \cueDuring "clarinet" #DOWN { c4. g8 }
g4 bes4
}

pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  \<<
    \new PianoStaff <<
    \new Staff {
      \new Voice {
        \pianoRH
      }
    }
    \new Staff {
      \clef "bass"
      \pianoLH
    }
    >>
  >>
}\score
```

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.
Selected Snippets

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
  (parser location 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
}
>>
>>

here's the lyrics
Flute
Clar.

See also

Notation Reference: Section 5.5.1 [Aligning objects], page 591, Section 5.4.2 [Direction and placement], page 577, [Formatting cue notes], page 198, [Quoting other voices], page 195, Section 5.6 [Using music functions], page 604.

Snippets: Section “Vocal music” in Snippets.

Internals Reference: Section “InstrumentSwitch” in Internals Reference, Section “CueVoice” in Internals Reference.

Known issues and warnings
\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 [Special note heads], page 34.
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:

\begin{verbatim}
a4 \markup { \smallCaps { Alex - } \italic { He's gone } } a a a
da4 a a \markup { \smallCaps { Bethan - } \italic { Where? } } a
da4 a a a
\end{verbatim}

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.

\begin{verbatim}
\score { <<
\new Lyrics \with {
\override LyricText.font-shape = #'italic
\override LyricText.self-alignment-X = #LEFT
} 
{ \dialogue }
\new Staff { 
\new Voice { \music }
}
>>}
\end{verbatim}
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 Section 2.9 [Ancient notation], page 404.

See also

Notation Reference: [Manual syllable durations], page 245, Section 1.8 [Text], page 215.
Internal Reference: Section “LyricText” in Internals Reference.

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.

```
\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:
Bar lines can also be removed on a staff-by-staff basis:

```latex
\score {
\new StaffGroup <<
\new Staff {
  \relative c'' {
    a4 b c2 |
    a4 b c2 |
    a4 b c2 |
  }
}
\new Staff {
  \relative c'' {
    a4 b c2 |
    a4 b c2 |
    a4 b c2 |
  }
}
layout 
context {
  \Staff
  \remove "Bar_engraver"
}
}
```

```
\score {
\new ChoirStaff <<
\new Staff 
\with { \remove "Bar_engraver" } {
  \relative c'' {
    a4 b c2 |
    a4 b c2 |
    a4 b c2 |
  }
}
\new Staff {
  \relative c'' {
    a4 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.

```
a4 b c2 |
cadenzaOn
a4 b c2
\bar ""
a4 b c2
\cadenzaOff
a4 b c2 |
a4 b c2 |
```

Rests or pauses in chants can be indicated by modified bar lines.

```
a4
\cadenzaOn
b c2
a4 b c2
\bar ""
a4 b c2
a4 b c2
\bar ";"
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
}
```

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.
Chants usually omit the time signature and often omit the clef too.
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.

```latex
SopranoMusic = \relative g' { 
g1 | c2 b | a1 | \bar "||" 
a1 | d2 c | c b | c1 | \bar "||"
}

AltoMusic = \relative c' { 
e1 | g2 g | f1 | 
f1 | f2 e | d d | e1 |
}

TenorMusic = \relative a { 
c1 | c2 c | c1 | 
d1 | g,2 g | g g | g1 |
}

BassMusic = \relative c { 
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 
>>
} }
Some other approaches to setting such a chant are shown in the first of the following snippets.

**Selected Snippets**

*Chant or psalms notation*

This form of notation is used for the chant of the Psalms, where verses aren’t always the same length.

```latex
\score {
    \new Staff \with { \remove "Time_signature_engraver" }
    \stemOff = \hide Staff.Stem
    \stemOn = \undo \stemOff
}
```
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.

```latex
\include "gregorian.ly"

chant = \relative c' {
  \set Score.timing = ##f
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4(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
    }
  }
```
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 { O 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 Section 1.8.3 [Fonts], page 236.
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph #"scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \left-align {
        \line { O 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 salvation.

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 salvation.

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.
Today if ye will hear his voice *

harden | not your | hearts: as in the provocation * and as in the day of temptation |
-ation | in the | wilderness.

In other psalters an accent is placed over the syllable to indicate stress.
O come let us sing unto the Lord: let us heartily rejoice in the strength of our salvation.

The use of markup to center text, and arrange lines in columns is described in Section 1.8.2 [Formatting text], page 223.

Most of these elements are shown in one or other of the two verses in the template, see Section “Psalms” in Learning Manual.

See also


Notation Reference: Section 1.8.3 [Fonts], page 236, Section 1.8.2 [Formatting text], page 223.

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"
        }
      }
    }
    ""
  }
}
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 Section 2.9 [Ancient notation], page 404.

See also

Notation Reference: Section 2.9 [Ancient notation], page 404.

2.2 Keyboard and other multi-staff instruments
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], page 162.
• Keyboard music can be written in parallel, as described in [Writing music in parallel], page 172.
• 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], page 114.
• Fingerings are indicated with [Fingering instructions], page 205.
• Organ pedal indications are inserted as articulations, see Section A.13 [List of articulations], page 699.
• Vertical grid lines can be shown with [Grid lines], page 212.
• Keyboard music often contains Laissez vibrer ties as well as ties on arpeggios and tremolos, described in [Ties], page 49.
• Placing arpeggios across multiple voices and staves is covered in [Arpeggio], page 133.
• Tremolo marks are described in [Tremolo repeats], page 152.
• Several of the tweaks that can occur in keyboard music are demonstrated in Section “Real music example” in Learning Manual.
• Hidden notes can be used to produce ties that cross voices, as shown in Section “Other uses for tweaks” in Learning Manual.

See also


Notation Reference: [Grouping staves], page 176, [Instrument names], page 192, [Collision resolution], page 162, [Writing music in parallel], page 172, [Fingering instructions], page 205, Section A.13 [List of articulations], page 699, [Grid lines], page 212, [Ties], page 49, [Arpeggio], page 133, [Tremolo repeats], page 152.

Internals Reference: Section “PianoStaff” in Internals Reference.

Snippets: Section “Keyboards” in Snippets.

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 Section 5.1.3 [Keeping contexts alive], page 550.

Cross-staff notes are beamed automatically:

\new PianoStaff <<
\new Staff = "up" {
\change Staff = "up"
<\e' c'>8
\change Staff = "down"
g8 fis g
\change Staff = "up"
<g' c''>8
\change Staff = "down"
e8 dis e
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 Section “Fixing overlapping notation” in Learning Manual.

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 Section “Fixing overlapping notation” in Learning Manual.

See also


Notation Reference: [Stems], page 210, [Automatic beams], page 76, Section 5.1.3 [Keeping contexts alive], page 550.

Snippets: Section “Keyboards” in Snippets.


Known issues and warnings

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 
  }
}

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.

Note: 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 c' {
      g8 b a c b d c e
      d8 r fis, g a2
    }
  }
}
\new Staff = "down" {
  \key g \major
  \clef bass
}
>>

See also

Notation Reference: [Changing staff manually], page 303.
Snippets: Section “Keyboards” in Snippets.

Known issues and warnings

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
c1
  \change Staff = "two"
  b2 a
}
\new Staff = "two" {
  \clef bass
  s1*2
}
>>
Predefined commands
\showStaffSwitch, \hideStaffSwitch.

See also
Snippets: Section “Keyboards” in Snippets.
Internals Reference: Section “Note_head_line_engraver” in Internals Reference, Section “VoiceFollower” in Internals Reference.

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.
Selected Snippets

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 \bar "||"
  \note b\sevenths -\> cis-. e-. gis-. b-.)!\fermata\laissezVibrer
  \}
}
\new Staff {
  \relative c' {
  \clef bass \key b \major <<
  { \note <a e cis>2.\arpeggio 
  } \\ 
  { \note <a, e a,>2. 
  } >>
  }
}
>>

See also

Snippets: Section “Keyboards” in Snippets.

Internals Reference: Section “Stem” in Internals Reference.

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.

\[c_4\]sustainOn d e g
\(<c, f a>1\]sustainOff
\[c_4\]sostenutoOn e g c,
\(<b e s d f>1\]sostenutoOff
\[c_4\]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.

\[c_4\]sustainOn g c2\]sustainOff
\set Staff.pedalSustainStyle = #'mixed
\[c_4\]sustainOn g c d
d\]sustainOff\]sustainOn g, c2\]sustainOff
\set Staff.pedalSustainStyle = #'bracket
c\]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.

See also

Notation Reference: [Ties], page 49.
Snippets: Section “Keyboards” in Snippets.

2.2.3 Accordion

This section discusses notation that is unique to the accordion.
Discant symbols

Accordion 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.

Selected Snippets

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 standardised. The available commands can be found in Section “Accordion Registers” in Notation Reference.

\layout { ragged-right = ##t }

#(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" }
  >> }
>>
See also

Snippets: Section “Keyboards” in Snippets.

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], page 128.
- A *bisbigliando* is written as a tremelo [Tremolo repeats], page 152.
- Natural harmonics are covered under [Harmonics], page 313.
- For directional arpeggios and non-arpeggios, see [Arpeggio], page 133.

See also
Notation Reference: [Tremolo repeats], page 152, [Glissando], page 128, [Arpeggio], page 133, [Harmonics], page 313.

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
\cis1 \markup \concat \vcenter {
  \[D \flat C \sharp B|E \sharp F \sharp G A \flat]\}
\c1 \markup \concat \vcenter {
  [ C \natural ] }

or pedal diagrams:

\textLengthOn
\cis1 \markup \harp-pedal "^v-|vv-~"
\c1 \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.

See also
Notation Reference: [Text scripts], page 216, Section A.10.5 [Instrument Specific Markup], page 682.
2.3 Unfretted string instruments

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], page 216.
- Fingerings, including the thumb indication, are described in [Fingering instructions], page 205.
- Double stopping is normally indicated by writing a chord, see [Chorded notes], page 154. Directives for playing chords may be added, see [Arpeggio], page 133.
- Templates for string quartets can be found in Section “String quartet templates” in Learning Manual. Others are shown in the snippets.
Bowing indications

Bowing indications are created as articulations, which are described in [Articulations and ornamentations], page 111.

The bowing commands, \downbow and \upbow, are used with slurs as follows:
\begin{verbatim}
c4(\downbow d) e(\upbow f)
\end{verbatim}

and the following example shows three ways in which an open A string on a violin might be indicated:
\begin{verbatim}
a4 \open
a^\markup { \teeny "II" }
a2^\markup { \small "sul A" }
\end{verbatim}

Predefined commands

\downbow, \upbow, \open.

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.
\begin{verbatim}
d4 e4.
\harmonicsOn
d8 e e
d4 e4.
\harmonicsOff
d8 e e
\end{verbatim}

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:
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.

\begin{verbatim}
<e a\harmonic>2. <c g'\harmonic>4
\set harmonicDots = ##t
<e a\harmonic>2. <c g'\harmonic>4
\end{verbatim}

Note: \harmonic must be placed inside a chord construct even if there is only a single note. Normally \harmonicsOn would be used in this situation.

See also


Notation Reference: [Special note heads], page 34, [References for unfretted strings], page 312.

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.

\begin{verbatim}
c\snappizzicato
\end{verbatim}

\begin{verbatim}
<c' e g>4\snappizzicato
<c' e g>4\snappizzicato
<c, e g>4\snappizzicato
\end{verbatim}
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], page 205.
- Instructions for Laissez vibrer ties as well as ties on arpeggios and tremolos can be found in [Ties], page 49.
- Instructions for handling multiple voices can be found in [Collision resolution], page 162.
- Instructions for indicating harmonics can be found in [Harmonics], page 313.

See also

Notation Reference: [Fingering instructions], page 205, [Ties], page 49, [Collision resolution], page 162, [Instrument names], page 192, [Writing music in parallel], page 172, [Arpeggio], page 133, Section A.13 [List of articulations], page 699, [Clef], page 16, [Instrument transpositions], page 24.

String number indications

The string on which a note should be played may be indicated by appending \number to a note.

\clef "treble_8"
c\5 e\4 g\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"
g\3-0
g-0\3
<g\3-0>
<g-0\3>

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.
Selected Snippets

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>
\}

See also

Notation Reference: [Fingering instructions], page 205.

Snippets: Section “Fretted strings” in Snippets.

Internals Reference: Section “StringNumber” in Internals Reference, Section “Fingering” in Internals Reference.
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 c' {
  a,8 a' <c e> a
  d,8 a' <d f> a
}
```

Default tablatures do not contain any symbols for tone duration nor any other musical symbols such as e.g. expressive marks.

```
symbols = {
  \time 3/4
  c4-."Allegro" d( e)
  f4-./f g a"fermata
  \mark \default
  c8_.<( c16 c^c2!)
  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.
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.
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], page 156. This particular command is so common that it is available as \tabChordRepeats.

guitar = \relative c' {
  r8 <gis-2 cis-3 b-0>~ q4 q8~ q 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 c' {
  \repeat volta 2 {
    e2. f4~
    f2 g2~
  }
  \alternative {
    { g4 f2. }
    { g4 \repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}

\score {
The command `\hideSplitTiedTabNotes` cancels the behavior of engraving fret numbers in parentheses:

ties = \relative c' { \\
\repeat volta 2 { \\
e2. f4~ \\
f2 g2~ } \\
\alternative { \\
{ g4 f2. } \\
{ g4\repeatTie c,2. } \\
} \\
b1~ \\
\break \\
b1 \\
\bar "\-"
Harmonic indications can be added to tablature notation as sounding pitches:

```latex
\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 }
  \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.

\begin{verbatim}
\definemacro*{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 }
}
\end{verbatim}

Alternatively, harmonics can be computed by defining the ratio of string lengths above and below the harmonic fingering.

\begin{verbatim}
\definemacro*{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 }
}
\end{verbatim}
Selected Snippets

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.

```latex
\new TabStaff {
  \relative c {
    \tabFullNotation
    g16 b d g b d g b
    \stemDown
    \override Beam.concaveness = #10000
    g,,16 b d g b d g b
  }
}
```

Polyphony in tablature

Polyphony is created the same way in a TabStaff as in a regular staff.

```latex
upper = \relative c' {
  \time 12/8
  \key e \minor
  \voiceOne
  r4. r8 e, fis g16 b g e e' b c b a g fis e
}

lower = \relative c {
  \key e \minor
  \voiceTwo
  r16 e d c b a g4 fis8 e fis g a b c
}
```

```latex
\score {
<<
  \new StaffGroup = "tab with traditional" <<
  \new Staff = "guitar traditional" <<
  \clef "treble_8"
  \context Voice = "upper" \upper
  \context Voice = "lower" \lower
```
Open string harmonics in tablature

This snippet demonstrates open-string harmonics

\begin{verbatim}
openStringHarmonics = {
  %first harmonic
  \harmonicByFret #12 e,2,6,\markup{"1st harm."}
  \harmonicByRatio #1/2 e,6
  %second harmonic
  \harmonicByFret #7 e,6,\markup{"2nd harm. -- -- --"}
  \harmonicByRatio #1/3 e,6
  \harmonicByFret #19 e,6
  \harmonicByRatio #2/3 e,6
  \harmonicByFret #19 < e,6 a,5 d\4 >
  \harmonicByRatio #2/3 < e,6 a,5 d\4 >
  %third harmonic
  \harmonicByFret #5 e,6,\markup{"3rd harm. -- -- --"}
  \harmonicByRatio #1/4 e,6
  \harmonicByFret #24 e,6
  \harmonicByRatio #3/4 e,6
  \break
  %fourth harmonic
  \harmonicByFret #4 e,6,\markup{"4th harm. -- -- -- -- -- -- -- -- --"}
  \harmonicByRatio #1/5 e,6
  \harmonicByFret #9 e,6
  \harmonicByRatio #2/5 e,6
  \harmonicByFret #16 e,6
  \harmonicByRatio #3/5 e,6
  %fifth harmonic
  \harmonicByFret #3 e,6,\markup{"5th harm."}
  \harmonicByRatio #1/6 e,6
  \break
  %sixth harmonic
  \harmonicByFret #2.7 e,6,\markup{"6th harm."}
  \harmonicByRatio #1/7 e,6
\end{verbatim}

\end{verbatim}
Fretted-string harmonics in tablature

Demonstrates fretted-string harmonics in tablature

\textHarmonicByFret #4 g4\3
\textHarmonicByFret #5 d4\4
\textHarmonicByFret #7 g2\3
% pinched harmonics (PH)
\textHarmonicByFret #7 d2\4
\textHarmonicByFret #5 d4\4
\textHarmonicByFret #7 a4\5
% tapped harmonics (TH)
\textHarmonicByFret #5 d4\4
\textHarmonicByFret #7 d4\4
\textHarmonicByFret #5 d4\4
\textHarmonicByFret #7 a4\5
% touched harmonics (TCH)
\textHarmonicByFret #9 g2.\3
\score {
  <<
    \new Staff {
      \new Voice {
        \clef "treble_8"
        \harmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \frettedStrings
      }
    }
  }
  >>
}

Slides in tablature

Slides can be typeset in both Staff and TabStaff contexts:

slides = {
  c'8\3(\glissando d'8\3)
  c'8\3\glissando d'8\3
  \hideNotes
  \grace { g16\glissando }
  \unHideNotes
  c'4\3
  \afterGrace d'4\3\glissando {
    \stemDown \hideNotes
g16 }
  \unHideNotes
}

\score {
  <<
    \new Staff { \clef "treble_8" \slides }
    \new TabStaff { \slides }
  }
  >>
  \layout {
    \context {
      \Score
      \override Glissando.minimum-length = #4
      \override Glissando.springs-and-rods = 
        #ly:spanner::set-spacing-rods
Chord glissando in tablature

Slides for chords can be indicated in both Staff and TabStaff. String numbers are necessary for TabStaff because automatic string calculations are different for chords and for single notes.

\begin{music}
\myMusic = \relative c' {<c\3 e\2 g\1>1 \glissando <f\3 a\2 c\1>}
\end{music}

\begin{music}
\score {<<
\new Staff {\clef "treble_8" \myMusic
}\new TabStaff {\myMusic
>>}
}
\end{music}

Hammer on and pull off

Hammer-on and pull-off can be obtained using slurs.

\begin{music}
\new TabStaff {
\relative c' {
\d4( e\2)
a( g)
}
}
\end{music}
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)
  }
}

See also

Notation Reference: [Chord repetition], page 156, [Glissando], page 128, [Harmonics], page 313, [Stems], page 210, [Written-out repeats], page 148.

Snippets: Section “Fretted strings” in Snippets.

Internals Reference: Section “TabNoteHead” in Internals Reference, Section “TabStaff” in Internals Reference, Section “TabVoice” in Internals Reference, Section “Beam” in Internals Reference.

Known issues and warnings

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
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, {
  c4 d e f
}
}
\new TabStaff \with {
  stringTunings = #bass-tuning
} {
\relative c, {
  c4 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], page 1. 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':

\begin{verbatim}
mynotes = {
c'4 e' g' c'' |
e''4 g'' b'' c'''
}
<<
\new Staff {
  \clef treble
  \mynotes
}
\new TabStaff {
  \set TabStaff.stringTunings = \stringTuning <c' g' d'' a''>
  \mynotes
}
>>
\end{verbatim}

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], page 343).

The previous example could also be written as follows:
custom-tuning = \stringTuning <c' g' d'' a''>

\begin{verbatim}
mynotes = {
c'4 e' g' c'' |
e''4 g'' b'' c'''
}
<<
\new Staff {
  \clef treble
  \mynotes
}
\new TabStaff {
  \set TabStaff.stringTunings = #custom-tuning
  \mynotes
}
>>
\end{verbatim}
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 \( \text{ly:make-pitch} \) (see Section A.21 [Scheme functions], page 747).

\( \text{stringTuning} \) 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

\[\text{layout { } context { } TabStaff{ } stringTunings = \text{stringTuning <c' g' d' a'>}}\]

A modern tab clef can also be used.

\[\text{new TabStaff { } clef moderntab{ } <a, e a>1{ } break{ } clef tab{ } <a, e a>1}\]

The modern tab clef supports tablatures from 4 to 7 strings.

See also

Notation Reference: [Absolute octave entry], page 1, [Predefined fret diagrams], page 343, Section A.21 [Scheme functions], page 747.


Snippets: Section “Fretted strings” in Snippets.

Internals Reference: Section “Tab_note_heads_ engraver” in Internals Reference.

Known issues and warnings

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 Section A.10.5 [Instrument Specific Markup], page 682.

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;"
  }
}
>>

\begin{music}
\begin{musicstaffs}
  \set {\clef = "treble_8"}
  \set {\timesig = \c44} \\
  <c e g c' e'> \fret-diagram #"6-x;5-3;4-2;3-o;2-1;1-o;"
\end{musicstaffs}
\end{music}

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;4-2;2-1;1-1;"
  }
  <g, d g b d' g'>1\markup {
    \fret-diagram #"c:6-1-3;6-3;5-2;5-4;4-3;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.

\begin{music}
\new ChordNames { 
  \chordmode { 
    c1 d:m 
  } 
}\end{music}

\begin{music}
\new Staff { 
  \clef "treble_8" 
  \chord { 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;" 
  }} 
  \chord { 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;" 
  }} 
}\end{music}

Dot radius and dot position can be controlled with the fret-diagram markup string.

\begin{music}
\new ChordNames { 
  \chordmode { 
    c1 d:m 
  } 
}\end{music}

\begin{music}
\new Staff { 
  \clef "treble_8" 
  \chord { c e g c' e'>1 \markup { 
    \fret-diagram "d:0.35;6-x;5-3;4-2-3-o;2-1;1-o;" 
  }} 
  \chord { d a d' f'>1 \markup { 
    \fret-diagram "p:0.2;6-x;5-x;4-o;3-2-2-3;1-1-1;" 
  }} 
}\end{music}
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.

Barre indicators can be included in the fret-diagram-terse markup string.
Fingering indications can be included in the fret-diagram-terse markup string.

```latex
<< \new ChordNames { \chordmode { cl d:m } } \new Staff { \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;0;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.

```latex
<< \new ChordNames { \chordmode { cl 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) } } >>
```
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.
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 Section “fret-diagram-interface” in Internals Reference. For a fret diagram markup, the interface properties belong to Voice.TextScript.

**Selected Snippets**

*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
  }
}>>
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)
%% 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;"
  }
}
}

>>

\begin{figure}
\centering
\includegraphics{guitar_chords.png}
\end{figure}
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`. 
Chord pitches can be entered either as simultaneous music or using chord mode (see [Chord mode overview], page 384).

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.
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
}
>>

C F G E A B

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 Section A.4 [Predefined fretboard diagrams], page 613. 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], page 353.

\include "predefined-guitar-fretboards.ly"
mychords = \chordmode{
  c1 c:maj9
}

<<
\new ChordNames {
  mychords
}
\new FretBoards {
  mychords
}
>>

C C\triangle^9
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;"

mychords = \chordmode {
  c1 c:maj9
}

<<
  \new ChordNames {
    mychords
  }
  \new FretBoards {
    mychords
  }
>>

C cmaj9

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 Section “fret-diagram-interface” in Internals Reference. For a predefined fret diagram, the interface properties belong to FretBoards.FretBoard.

Selected Snippets

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 = #\(,(\text{ly:make-pitch} 0 \ 6 \ 0)
\),\(\text{ly:make-pitch} 1 \ 3 \ \text{SHARP})
\),\(\text{ly:make-pitch} 1 \ 1 \ 0)
\),\(\text{ly:make-pitch} 0 \ 5 \ 0)\)

dSix = \{ \langle a\4 b\1 d\3 fis\2\rangle \}
dMajor = \{ \langle a\4 d\1 d\3 fis \2\rangle \}
aMajSeven = \{ \langle a\4 cis\1 e\3 g\2\rangle \}
dMajSeven = \{ \langle a\4 cis\1 d\3 fis\2\rangle \}
gMajor = \{ \langle b\4 b\1 d\3 g\2\rangle \}

\storePredefinedDiagram #default-fret-table \dSix
\storePredefinedDiagram #default-fret-table \dMajor
\storePredefinedDiagram #default-fret-table \aMajSeven
\storePredefinedDiagram #default-fret-table \dMajSeven
\storePredefinedDiagram #default-fret-table \gMajor

% 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 {
  \context {
    \Score
    \override SpacingSpanner.base-shortest-duration = #(ly:make-moment 1/16)
  }
}

\midi { }
**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 |
  }
  >>
}
>>
See also

Notation Reference: [Custom tablatures], page 331, [Automatic fret diagrams], page 353, [Chord mode overview], page 384, Section A.4 [Predefined fretboard diagrams], page 613.


 Snippets: Section “Fretted strings” in Snippets.

 Internals Reference: Section “fret-diagram-interface” in Internals Reference.

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:
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.
The minimum fret to be used in calculating strings and frets for the FretBoard context can be set with the `minimumFret` property.

```latex
\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], page 331 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 Section “fret-diagram-interface” in Internals Reference. For a FretBoards fret diagram, the interface properties belong to `FretBoards.FretBoard`.

**Predefined commands**

`\predefinedFretboardsOff`, `\predefinedFretboardsOn`. 
See also

Notation Reference: [Custom tablatures], page 331.
Snippets: Section “Fretted strings” in Snippets.
Internals Reference: Section “fret-diagram-interface” in Internals Reference.

Known issues and warnings

Automatic fretboard calculations do not work properly for instruments with non-monotonic tunings.

Right-hand fingerings

Right-hand fingerings \textit{p-i-m-a} must be entered using \texttt{\rightHandFinger} followed by a number.

\begin{itemize}
  \item \textbf{Note:} If the number is entered in Scheme notation, remember to append a space before following it with a closing \textgreater or similar.
\end{itemize}

\begin{verbatim}
\clef "treble_8"
c\rightHandFinger #1
e\rightHandFinger #2
g\rightHandFinger #3
c\rightHandFinger #4
<\text{c,-} \rightHandFinger #1 \text{e,-} \rightHandFinger #2
g\rightHandFinger #3 c\rightHandFinger #4 >1
\end{verbatim}

For convenience, you can abbreviate \texttt{\rightHandFinger} to something short, for example \texttt{RH}, \texttt{RH=#rightHandFinger}

Selected Snippets

\textit{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

\begin{verbatim}
#(define RH rightHandFinger)
\relative c {
  \clef "treble_8"
  \set strokeFingerOrientations = #'(up down)
  <\text{c,-} \text{RH} #1 \text{e,-} \text{RH} #2 \text{g,-} \text{RH} #3 \text{c,-} \text{RH} #4 >4

  \set strokeFingerOrientations = #'(up right down)
  <\text{c,-} \text{RH} #1 \text{e,-} \text{RH} #2 \text{g,-} \text{RH} #3 \text{c,-} \text{RH} #4 >4

  \set strokeFingerOrientations = #'(left)
  <\text{c,-} \text{RH} #1 \text{e,-} \text{RH} #2 \text{g,-} \text{RH} #3 \text{c,-} \text{RH} #4 >2
}
\end{verbatim}
Fingerings string indications and right-hand fingerings

This example combines left-hand fingering, string indications, and right-hand fingering.

```lilypond
#(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
}
```

See also

Snippets: Section “Fretted strings” in Snippets.
Internals Reference: Section “StrokeFinger” in Internals Reference.

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.

```lilypond
\clef "treble_8"
b16 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
```

See also

Notation Reference: [Text spanners], page 217.
Snippets: Section “Fretted strings” in Snippets, Section “Expressive marks” in Snippets.
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 c' { 
\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 c' { 
< 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
}

\begin{music}
\times 4\times 
\new ChordNames {
\ChordsAndSymbols
}
\new Staff {
\clef "treble_8"
\ChordsAndSymbols
}
\new TabStaff {
\ChordsAndSymbols
}
\end{music}

See also

Snippets: Section “Fretted strings” in Snippets.
Notation Reference: [Special note heads], page 34, Section A.9 [Note head styles], page 645.

Indicating power chords

Power chords and their symbols can be engraved in chord mode or as chord constructs:

\begin{music}
\times 4\times 
\new ChordNames {
\ChordsAndSymbols
}
\new Staff {
\clef "treble_8"
\ChordsAndSymbols
}
\end{music}
Power chord symbols are automatically switched off as soon as one of the other common chord modifier is used:

```latex
mixedChords = \chordmode {
  c,1
  \powerChords
  b,,i:1.5
  fis,,i:1.5.8
  g,,i:m
}
\score {
  \new ChordNames {
    mixedChords
  }
  \new Staff {
    \clef "treble_8"
    \mixedChords
  }
  \new TabStaff {
    \mixedChords
  }
}
```

See also

Music Glossary: Section “power chord” in Music Glossary.
Notation Reference: [Extended and altered chords], page 387, [Printing chord names], page 389.
Snippets: Section “Fretted strings” in Snippets.

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:

```lilypond
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:

```lilypond
\set TabStaff.stringTunings = #(four-string-banjo banjo-c-tuning)
```

See also

- Installed Files: 'ly/string-tunings-init.ly'.
- Snippets: Section “Fretted strings” in Snippets.

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], page 73, and [Instantiating new staves], page 175.
- MIDI output is discussed in a separate section; please see Section 3.5.6 [Percussion in MIDI], page 491.
See also

Notation Reference: [Showing melody rhythms], page 73, [Instantiating new staves], page 175. Section 3.5.6 [Percussion in MIDI], page 491.

Snippets: Section “Percussion” in Snippets.

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 Section A.14 [Percussion notes], page 700.

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 bd bd bd
    \clef treble
    hh4 hh hh hh
}

There are a few issues concerning MIDI support for percussion instruments; for details please see Section 3.5.6 [Percussion in MIDI], page 491.

See also

Notation Reference: Section 3.5.6 [Percussion in MIDI], page 491, Section A.14 [Percussion notes], page 700.


Snippets: Section “Percussion” in Snippets.
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], page 152.

\drums {
  \time 2/4
  sn16 sn8 sn16 sn8 sn8:32 ~
  sn8 sn8 sn4:32 ~
  sn4 sn8 sn16 sn16
  sn4 r4
}

Sticking can be indicated by placing a markup for "R" or "L" above or below notes, as discussed in Section 5.4.2 [Direction and placement], page 577. The staff-padding property may be overridden to achieve a pleasing baseline.

\drums {
  \repeat unfold 2 {
    sn16="L" sn="R" sn="L" sn="L" sn="L" sn="R" sn="R" sn="R"
    \stemUp
    sn16="L" sn="R" sn="L" sn="L" sn="L" sn="R" sn="R" sn="R"
  }
}

See also

Notation Reference: [Tremolo repeats], page 152.
Snippets: Section “Percussion” in Snippets.

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.

See also

Notation Reference: Section 3.5.6 [Percussion in MIDI], page 491.
Snippets: Section “Percussion” in Snippets.

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.
The above example shows verbose polyphonic notation. The short polyphonic notation, described in Section “I’m hearing Voices” in Learning Manual, 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:
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.

timbales-style
This typesets timbales on a two line staff:

\[ \text{timh ssh timl ssl cb} \]

congas-style
This typesets congas on a two line staff:

\[ \text{cgh cgho cghm ssh cgl cglo cglm ssl} \]

bongos-style
This typesets bongos on a two line staff:

\[ \text{boh boho bohm ssl bol bol bolm ssl} \]

percussion-style
To typeset all kinds of simple percussion on one line staves:

\[ \text{tri trio trim guis guil cb cl tamb cab mar hc} \]

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.

\[ \begin{align*}
#(\text{define mydrums }'(\text{bassdrum \ default \ #f \ -1)}\
\text{(snare \ default \ #f \ 0)}\
\text{(hihat \ cross \ #f \ 1)}\
\text{(halfopenhihat \ cross \ "halfopen" \ 1)}\
\text{(pedalhihat \ xcircle \ "stopped" \ 2)}\
\text{(lowtom \ diamond \ #f \ 3)})\end{align*} \]

up = \drummode { hh8 hh hhho hhho hhp4 hhp } down = \drummode { bd4 sn bd toml8 toml } \new DrumVoice { \voiceOne \up } \new DrumVoice { \voiceTwo \down }
Selected Snippets

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)
    (lowoodblock 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 wbl16 wbl wblwbl8-> wbl16 wbl
    wbl16 wbl16 wbl16 wbl16 r16
  }
}

Note that in this special case the length of the barline must 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], page 182.

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. tamb16 tamb tamb tamb tamb |
  tamb4. tamb8 tamb tamb |
  \% 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 |
}

Tambourine
\[ \begin{array}{c}
  \text{\includegraphics[width=0.5\textwidth]{tambourine.png}}
\end{array} \]

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
}

Tamtam
\[ \begin{array}{c}
  \text{\includegraphics[width=0.5\textwidth]{tamtam.png}}
\end{array} \]

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"
Here a short example taken from Stravinsky’s ‘L’histoire du Soldat’.

```scheme
#(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
}
```

```latex
\new DrumStaff { 
  \bellstaff 
  \drummode { 
    \time 2/4 
    rb8 rb cb cb16 rb-> ~ | 
    rb16 rb8 rb16 cb8 cb | 
  } 
}
```

Different Bells

Here a short example taken from Stravinsky’s ‘L’histoire du Soldat’.

See also

Snippets: Section “Percussion” in Snippets.

Internals Reference: Section “DrumStaff” in Internals Reference, Section “DrumVoice” in Internals Reference.

Ghost notes

Ghost notes for drums and percussion may be created using the \parenthesize command detailed in [Parentheses], page 209.

\new DrumStaff <<
\context DrumVoice = "1" { s1 }
\context DrumVoice = "2" { s1 }
\drummode {
<<
{ hh8[ hh] <hh sn> hh16
\parenthesize sn hh
\parenthesize sn hh8 <hh sn> hh
} \ \\ 
{ bd4 r4 bd8 bd r8 bd }
} >>
See also

Snippets: Section “Percussion” in Snippets.

2.6 Wind instruments

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], page 126.
- Legato playing is indicated by [Slurs], page 122.
- 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], page 111 and Section A.13 [List of articulations], page 699.
- Flutter tonguing is usually indicated by placing a tremolo mark and a text markup on the note. See [Tremolo repeats], page 152.

Other aspects of musical notation that can apply to wind instruments:

- Many wind instruments are transposing instruments, see [Instrument transpositions], page 24.
- Slide glissandi are characteristic of the trombone, but other winds may perform keyed or valved glissandi. See [Glissando], page 128.
- Harmonic series glissandi, which are possible on all brass instruments but common for French Horns, are usually written out as [Grace notes], page 104.
- Pitch inflections at the end of a note are discussed in [Falls and doits], page 127.
- Key slaps or valve slaps are often shown by the cross style of [Special note heads], page 34.
Woodwinds can overblow low notes to sound harmonics. These are shown by the \texttt{flageolet} articulation. See Section A.13 [List of articulations], page 699.

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 \texttt{stopped} and \texttt{open} articulations. See [Articulations and ornamentations], page 111 and Section A.13 [List of articulations], page 699.

Stopped horns are indicated by the \texttt{stopped} articulation. See [Articulations and ornamentations], page 111.

\section*{Selected Snippets}

\textit{Changing \texttt{flageolet} mark size}

To make the \texttt{flageolet} circle smaller use the following Scheme function.

\begin{verbatim}
smallFlageolet = 
  #(let ((m (make-articulation "flageolet")))
    (set! (ly:music-property m 'tweaks)
      (acons 'font-size -3
        (ly:music-property m 'tweaks)))
    m)
\end{verbatim}

See also

Notation Reference: [Breath marks], page 126, [Slurs], page 122, [Articulations and ornamentations], page 111, Section A.13 [List of articulations], page 699, [Tremolo repeats], page 152, [Instrument transpositions], page 24, [Glissando], page 128, [Grace notes], page 104, [Falls and doits], page 127, [Special note heads], page 34.

Snippets: Section “Winds” in Snippets.

\section*{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 Section 2.6.3.1 [Woodwind diagrams], page 375.

\section*{Selected Snippets}

\textit{Fingering symbols for wind instruments}

Special symbols can be achieved by combining existing glyphs, which is useful for wind instruments.

\begin{verbatim}
centermarkup = {
  \once \override TextScript.self-alignment-X = #CENTER
\end{verbatim}
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 = #(ly:make-simple-closure
\(+
\,(ly:make-simple-closure (list
\ ly:self-alignment-interface::centered-on-x-parent))
\,(ly:make-simple-closure (list
\ ly:self-alignment-interface::x-aligned-on-self))))
}
\score {
\new Staff \with {
\remove "Time_signature_engraver"
\omit Stem
\omit Flag
\consists "Horizontal_bracket_engraver"
}
{\clef bass
\set Score.timing = ##f
f,1\cdot1/4 \glissando
\clef violin
}
See also

Notation Reference: Section 2.6.3.1 [Woodwind diagrams], page 375.

Snippets: Section “Winds” in Snippets.

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"
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 cflat or fflat. Similarly, the piobaireachd high g can be written gflat when it occurs in light music.

See also

Snippets: Section “Winds” in Snippets.

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
  \thrd 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 \hdblf f8[ e32 d16.]
  \grg f2 \grg f8 e
  \thrd d2 \grg b4
  \grG a2 \grg a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 e4
  \thrd d2.}
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:

\slur d2
\bar "|.

See also

Snippets: Section “Winds” in Snippets.
Keys can be open, partially-covered, ring-depressed, or fully covered:

\textLengthOn

c1``\markup {
\textLengthOn
\center-column {
"one quarter"
\woodwind-diagram #'flute '((cc . (one1q))
(1h . ()))
(rh . ()))
}
}

c1``\markup {
\center-column {
"one half"
\woodwind-diagram #'flute '((cc . (one1h))
(1h . ()))
(rh . ()))
}
}

c1``\markup {
\center-column {
"three quarter"
\woodwind-diagram #'flute '((cc . (one3q))
(1h . ()))
(rh . ()))
}
}

c1``\markup {
\center-column {
"ring"
\woodwind-diagram #'flute '((cc . (oneR))
(1h . ()))
(rh . ()))
}
}
Trills are indicated as shaded keys:

A variety of trills can be displayed:
Chapter 2: Specialist notation

\center-column { "ring to shut"
  \woodwind-diagram #'flute '((cc . (oneTR))
  (lh . ()
  (rh . ()))
}

\center-column { "ring to open"
  \woodwind-diagram #'flute '((cc . (oneRT))
  (lh . ()
  (rh . ()))
}

\center-column { "open to shut"
  \woodwind-diagram #'flute '((cc . (oneT))
  (lh . ()
  (rh . ()))
}

\center-column { "one quarter to three quarters"
  \woodwind-diagram #'flute '((cc . (one1qT3q))
  (lh . ()
  (rh . ()))
}

\timesig.C44
\clefs.G

one quarter to ring to ring to shut to ring to open

\noteheads.s0
\noteheads.s0
\noteheads.s0
\noteheads.s0
Chapter 2: Specialist notation

The list of all possible keys and settings for a given instrument can be displayed on the console using 
\( \text{(print-keys-verbose 'flute)} \) or in the log file using \( \text{(print-keys-verbose 'flute (current-error-port))} \), 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'.

Predefined commands

Selected Snippets

Woodwind diagrams listing

The following music shows all of the woodwind diagrams currently defined in LilyPond.

\relative c' {  
\textLengthOn  
c1\markup {  
\center-column {  
'tin-whistle  
" "  
\woodwind-diagram  
#'tin-whistle  
#'()  
} 
} 
\textLengthOn  
c1\markup {  
\center-column {  
'piccolo  
" "  
\woodwind-diagram  
#'piccolo  
#'()  
} 
} 
\textLengthOn  
c1\markup {  
\center-column {  
'flute  
" "  
\woodwind-diagram  
#'flute  
#'()  
} 
}
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.
Chapter 2: Specialist notation

Changing the size of woodwind diagrams

The size and thickness of woodwind diagrams can be changed.

\relative c'' {
\textLength0n
c\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.

```lisp
#(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))
```

See also


Snippets: Section “Winds” in Snippets.

2.7 Chord notation

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], page 154.

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 Section 5.4.1 [Input modes], page 575.

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:

\chordmode { c1 g a g c }
<c e g>2 <g’ b d>
\chordmode { f2 g }

\chordmode { c2 f }
<c e g>2 <g’ b d>

\chordmode { c2 f }
<c e g>2 <g’ b d>

\chordmode { c2 f g }

\chordmode { c2:m f4:aug g:dim }

\chordmode { c2:m f4:aug g:dim }

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:
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 Section A.2 [Common chord modifiers], page 608.

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Action</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>The default action; produces a major triad.</td>
<td>(\text{\texttt{\textbackslash{chordmode} { c1:7 c:m7 c:maj7 c:dim7 c:aug7 }}})</td>
</tr>
<tr>
<td>m, m7</td>
<td>The minor chord. This modifier lowers the 3rd.</td>
<td></td>
</tr>
<tr>
<td>dim, dim7</td>
<td>The diminished chord. This modifier lowers the 3rd, 5th and (if present) the 7th step.</td>
<td></td>
</tr>
<tr>
<td>aug</td>
<td>The augmented chord. This modifier raises the 5th step.</td>
<td></td>
</tr>
<tr>
<td>maj, maj7</td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

See also

Notation Reference: Section A.2 [Common chord modifiers], page 608, [Extended and altered chords], page 387.

Snippets: Section “Chords” in Snippets.

Known issues and warnings

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], page 387.
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 
}

Note that both c:5 and c produce a C major triad.

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 {
  c1:5.6 c:3.7.8 c:3.6.13 
}

Added steps can be as high as desired.
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-
}\n
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
}\n
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. sus is equivalent to ^3; sus4 is equivalent to .4^3.

\chordmode {
c1:sus c:sus2 c:sus4 c:5.4^3
}\n
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 {
c1 c/g 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 {
  c1 c/g c/+g
}

Chord modifiers that can be used to produce a variety of standard chords are shown in Section A.2 [Common chord modifiers], page 608.

See also

Notation Reference: Section A.2 [Common chord modifiers], page 608.
Snippets: Section “Chords” in Snippets.

Known issues and warnings

Each step can only be present in a chord once. The following simply produces the augmented chord, since \+ is interpreted last.

\chordmode { c1:5.5-.5+ }

Only the second inversion can be created by adding a bass note. The first inversion requires changing the root of the chord.

\chordmode {
  c'1: c':/g e:6-3-^5 e:m6-^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:
Rests passed to a \ChordNames context will cause the \noChordSymbol markup to be displayed.

\chords { \ldots } is a shortcut notation for \new \ChordNames \chordmode { \ldots }.

\chords {
c2 f4.:m g8:maj7
}
Selected Snippets

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 }
}
```

```
\chords { c2 g:sus4 f e }
\relative c'' {
  a4 e c8 e r4
  b2 c4( d)
}
\addlyrics { One day this shall be free __ }
```

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 __ }
```

```
C  Fm  G^\triangle
```

Cm

3

Cm  D

One day this shall be free_
See also

Notation Reference: [Writing music in parallel], page 172.
Snippets: Section “Chords” in Snippets.
Internals Reference: Section “ChordNames” in Internals Reference, Section “ChordName” in Internals Reference, Section “Chord_name_engraver” in Internals Reference, Section “Volta_engraver” in Internals Reference, Section “Bar_engraver” in Internals Reference.

Known issues and warnings

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 Section “Literature list” in Essay). 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 Section A.1 [Chord name chart], page 607.

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:

```plaintext
default          E/D  Cm  B/B  B#/B#  Bb/Bb
german           E/d  Cm  H/h  H#/his  B/b
semi-german      E/d  Cm  H/h  H#/his  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.

```latex
\new ChordNames { 
  <c e g d'> % add9 
  \set additionalPitchPrefix = "add"
  <c e g d'> % add9
}

C\textsuperscript{9} C\textsuperscript{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.

```latex
\chords { 
  c4:7.9- c:7.9-/g 
  \set chordNoteNamer = \markup { "b" } 
  \break 
  c4:7.9- c:7.9-/g 
}

C\textsuperscript{b9} C\textsuperscript{b9}/G
```

**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`.

```latex
\chords { 
  c4:7.9- c:7.9-/g 
  \set chordNameSeparator = \markup { "/" } 
  \break 
  c4:7.9- c:7.9-/g 
}

C\textsuperscript{7\textsuperscript{b9}} C\textsuperscript{7\textsuperscript{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.

```latex
\chords { 
  c4:7.9- c:7.9-/g 
  \set slashChordSeparator = \markup { " over " } 
}
```
Chapter 2: Specialist notation

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.

Predefined commands
\whiteTriangleMarkup, \blackTriangleMarkup, \germanChords, \semiGermanChords, \italianChords, \frenchChords.

Selected Snippets
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\text{△} 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\text{△} | F^{7} | B_{b}^{7}

Volta below chords
By adding the Volta_engraver to the relevant staff, volte can be put under chords.

\score{
  <<
    \chords{

Changing chord separator

The separator between different parts of a chord name can be set to any markup.

\chords {
    c:7sus4
    \set chordNameSeparator = \markup { \typewriter | }
    c:7sus4
}

C\textsuperscript{7}sus4 C\textsuperscript{7}|sus4

See also

Notation Reference: Section A.1 [Chord name chart], page 607, Section A.2 [Common chord modifiers], page 608.

Essay on automated music engraving: Section “Literature list” in Essay.


Snippets: Section “Chords” in Snippets.

Known issues and warnings

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 }
}
2.7.3 Figured bass

Figured bass notation can be displayed.
Introduction to figured bass

LilyPond has support for figured bass, also called thorough bass or basso continuo:

\begin{verbatim}
<<
\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
  }
}
>>
\end{verbatim}

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.

See also

Snippets: Section “Chords” in Snippets.

Entering figured bass

\figuremode is used to switch the input mode to figure mode. More information on different input modes can be found at Section 5.4.1 [Input modes], page 575.

In figure mode, a group of bass figures is delimited by < and >. The duration is entered after the >.

\begin{verbatim}
\new FiguredBass {
  \figuremode {
    <6 4>2
  }
}
\end{verbatim}

Accidentals (including naturals) can be added to figures:

\begin{verbatim}
\figures {
  <7! 6+ 4--> <5++> <3-->
}
\end{verbatim}
Augmented and diminished steps can be indicated:
\figures {
    <6\+ 5/> <7/>
}
\accidental { 7 \+6 }
\accidental { 6 5 }

A backward slash through a figure (typically used for raised sixth steps) can be created:
\figures {
    <6> <6\>
}
\accidental { 6 6 }

Vertical spaces and brackets can be included in figures:
\figures {
    <[12 _!] 8 [6 4]>
}
\accidental { [12] 8 6 4 }

Any text markup can be inserted as a figure:
\figures {
    <\markup { \tiny \number 6 \super (1) } 5>
}
\accidental { 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
  d\textsf{4} d c c}
}\>>

\set figuredBassAlterationDirection = #RIGHT
<6\!+> <5+> <6 4-> r

The table below summarizes the figure modifiers available.

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Purpose</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>+, -, !</td>
<td>Accidentals</td>
<td>#7 \times5 #3 #6 #4</td>
</tr>
<tr>
<td>+, /</td>
<td>Augmented and diminished steps</td>
<td>#6 #5</td>
</tr>
<tr>
<td>\</td>
<td>Raised sixth step</td>
<td>#6</td>
</tr>
<tr>
<td>!</td>
<td>End of continuation line</td>
<td>#6</td>
</tr>
</tbody>
</table>

Predefined commands
\bassFigureExtendersOn, \bassFigureExtendersOff.

Selected Snippets

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
}

\relative c'' {
  c4 c'8 r8 c,4 c'
}
\new FiguredBass {
  \figuremode {
    <4>4 <10 6>8 s8
    <6 4>4 <6 4>
  }
}

\new Staff = "myStaff"
\figuremode {
  <4>4 <10 6>8 s8
  <6 4>4 <6 4>
}

\context Staff = "myStaff"
{4 10 6 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.

\relative c'' {
  c4 c'8 r8 c,4 c'
}
\new FiguredBass {
  \figuremode {
    <4>4 <10 6>8 s8
    <6 4>4 <6 4>
  }
}

%% Put notes on same Staff as figures
\context Staff = "myStaff"
{
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'
}

Predefined commands
\bassFigureStaffAlignmentDown, \bassFigureStaffAlignmentUp, \bassFigureStaffAlignmentNeutral.

See also
Snippets: Section "Chords" in Snippets.
Internals Reference: Section "BassFigure" in Internals Reference, Section "BassFigureAlignment" in Internals Reference, Section "BassFigureLine" in Internals Reference, Section "BassFigureBracket" in Internals Reference, Section "BassFigureContinuation" in Internals Reference, Section "FiguredBass" in Internals Reference.

Known issues and warnings
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
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], page 7.
- Non-standard key signatures are addressed in [Key signature], page 20.
- Contemporary practises in displaying accidentals are addressed in [Automatic accidentals], page 25.

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], page 59.
- Basic polyrhythmic notation is addressed in [Polyrhythmic notation], page 69.
- Feathered beams are addressed in [Feathered beams], page 89.
- Mensural bar lines (bar lines between staves only) are addressed in [Grouping staves], page 176.

Tuplets in contemporary music

Contemporary time signatures

Extended polyrhythmic 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
- *Instrumentation and Orchestration* by Alfred Blatter [Schirmer, 2nd ed. 1997]

Scores and musical examples

2.9 Ancient notation

Salve, Regina, mater misericordiae: Ad

Te clamamus, exules filii Hevae. Ad te spes...
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], page 411,
- [Mensural accidentals and key signatures], page 413,
- [Mensural rests], page 412,
- [Mensural clefs], page 409,
- [Gregorian clefs], page 417,
- [Mensural flags], page 412,
- [Mensural time signatures], page 410.

Some notational concepts are introduced specifically for ancient notation,

- [Custodes], page 408,
- [Divisiones], page 418,
- [Ligatures], page 407.


**See also**


Notation Reference: [Mensural note heads], page 411, [Mensural accidentals and key signatures], page 413, [Mensural rests], page 412, [Gregorian clefs], page 417, [Mensural flags], page 412, [Mensural time signatures], page 410, [Custodes], page 408, [Divisiones], page 418, [Ligatures], page 407.

### 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 **Petrucci** 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.

**See also**


### 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], page 416,
- [Mensural contexts], page 408.

See also


Notation Reference: [Gregorian chant contexts], page 416, [Mensural contexts], page 408.

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 c'' {  
[ 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], page 414 and [Gregorian square neume ligatures], page 420.

See also


Notation Reference: [White mensural ligatures], page 414, [Gregorian square neume ligatures], page 420.

Known issues and warnings

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:

\score {
\new MensuralVoice = "discantus" \relative c'' {
  \hide Score.BarNumber {
    c\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
}

\begin{figure}
\centering
\begin{tikzpicture}
\node at (0,0) {San -- ctus,};
\end{tikzpicture}
\end{figure}

\begin{figure}
\centering
\begin{tikzpicture}
\node at (0,0) {San -- -- -- ctus,};
\end{tikzpicture}
\end{figure}

\begin{figure}
\centering
\begin{tikzpicture}
\node at (0,0) {San -- ctus,};
\end{tikzpicture}
\end{figure}

See also

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], page 16. The note printed to the right side of each clef in the example column denotes the c' with respect to that clef.

Petruchio used C clefs with differently balanced left-side vertical beams, depending on which staff line it is printed.

<table>
<thead>
<tr>
<th>Description</th>
<th>Supported Clefs</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>mensural C clef</td>
<td>mensural-c1, mensural-c2, mensural-c3, mensural-c4, mensural-c5</td>
<td></td>
</tr>
<tr>
<td>mensural F clef</td>
<td>mensural-f</td>
<td></td>
</tr>
<tr>
<td>mensural G clef</td>
<td>mensural-g</td>
<td></td>
</tr>
</tbody>
</table>
black mensural C clef

neomensural C clef

petrucci style C clefs, for use on different staff lines (the example shows the 2nd staff line C clef)

petrucci style F clefs, for use on different staff lines (the example shows the 3rd staff line F clef)

petrucci style G clef

See also

Notation Reference: [Clef], page 16.

Known issues and warnings
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
\O \O

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], page 59, gives a general introduction to the use of time signatures.

See also
Notation Reference: [Time signature], page 59.

Known issues and warnings
Ratios of note durations cannot change with the time signature, as those are not constant. For example, the ratio of 1 breve = 3 semibreves (tempus perfectum) can be made by hand, by setting

\begin{verbatim}
breveTP = #(ly:make-duration -1 0 3/2)
\end{verbatim}

...{ c\breveTP f1 }
This sets \texttt{breveTP} to 3/2 times 2 = 3 times a whole note.

The \texttt{mensural68alt} and \texttt{neomensural68alt} symbols (alternate symbols for 6/8) are not addressable with \texttt{\time}. Use \texttt{\markup {\musicglyph "timesig.mensural68alt" }} instead.

Mensural note heads
For ancient notation, a note head style other than the \texttt{default} style may be chosen. This is accomplished by setting the \texttt{style} property of the \texttt{NoteHead} object to \texttt{baroque}, \texttt{neomensural}, \texttt{mensural}, \texttt{petrucci}, \texttt{blackpetrucci} or \texttt{semipetrucci}.

The \texttt{baroque} style differs from the \texttt{default} style by:
• Providing a \texttt{maxima} note head, and
• Using a square shape for \texttt{\breve} note heads.

The \texttt{neomensural}, \texttt{mensural}, and \texttt{petrucci} styles differ from the \texttt{baroque} style by:
• Using rhomboidal heads for semibreves and all smaller durations, and
• Centering the stems on the note heads.

The \texttt{blackpetrucci} style produces note heads usable in black mensural notation or coloratio sections in white mensural notation. Because note head style does not influence flag count, in this style a semiminima should be notated as a\texttt{8\*2}, not a\texttt{4}, otherwise it will look like a \texttt{minima}. The multiplier can be different if coloratio is used e.g. to notate triplets.

Use \texttt{semipetrucci} style to draw half-colored note heads (breves, longas and maximas).

The following example demonstrates the \texttt{petrucci} style:

\begin{verbatim}
\set Score.skipBars = ##t
\autoBeamOff
\override NoteHead.style = #'petrucci
a'\maxima a'\longa a'\breve a'1 a'2 a'4 a'8 a'16 a'
\override NoteHead.style = #'semipetrucci
a'\breve*5/6
\override NoteHead.style = #'blackpetrucci
a'8\*4/3 a'
\end{verbatim}
\override NoteHead.style = #'petrucci a' \longa

Section A.9 [Note head styles], page 645, gives an overview of all available note head styles.

See also


Notation Reference: Section A.9 [Note head styles], page 645.

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.

\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.

See also


Known issues and warnings

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 styles are classical, neomensural, and mensural. classical differs from the default style only in that the quarter rest looks like a horizontally mirrored 8th rest. The mensural and the neomensural styles mimic the appearance of rests in manuscripts and prints up to the 16th century.

The following example demonstrates the mensural and neomensural styles:
\set Score.skipBars = ##t
\override Rest.style = #'classical
r\longa"classical" r\breve r1 r2 r4 r8 r16 s \break
\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

\begin{music}
\set Staff.substaff=2
r16-8-4-2-1
\end{music}

There are no 32th and 64th rests specifically for the mensural or neo-mensural style. Instead, the rests from the default style will be taken.

**See also**

Notation Reference: [Rests], page 52.
Snippets: Section “Ancient notation” in *Snippets*.

**Known issues and warnings**
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. If called for, the natural sign will be taken from the vaticana style.

\begin{music}
\set Staff.substaff=2
r16-8-4-2-1
\end{music}

The style for accidentals and key signatures is controlled by the glyph-name-alist property of the grobs Accidental and KeySignature, respectively; e.g.:
\begin{verbatim}
\override Staff.Accidental.glyph-name-alist = #alteration-mensural-glyph-name-alist
\end{verbatim}
See also


Notation Reference: Section 1.1 [Pitches], page 1, [Accidentals], page 5, [Automatic accidentals], page 25, [Key signature], page 20.

Internals Reference: Section “KeySignature” in Internals Reference.

Annotational accidentals (\textit{musica ficta})

In European music from about 1600, singers were expected to chromatically alter notes at their own initiative according to certain rules. This is called \textit{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 \texttt{suggestAccidentals} to true.

\begin{verbatim}
fis gis
\set suggestAccidentals = ##t
ais bis
\end{verbatim}

This will treat every subsequent accidental as \textit{musica ficta} until it is unset with \texttt{suggestAccidentals = ##f}. A more practical way is to use \texttt{\once \set suggestAccidentals = ##t}, which can even be defined as a convenient shorthand:

\begin{verbatim}
ficta = { \once \set suggestAccidentals = ##t }
\score { \relative c''
\new MensuralVoice { 
\once \set suggestAccidentals = ##t 
bes4 a2 g2 \ficta fis8 \ficta e! fis2 g1 
} }
\end{verbatim}

See also

Internals Reference: Section “Accidental\_engraver” in Internals Reference, Section “Accidental\_Suggestion” in Internals Reference.

White mensural ligatures

There is limited support for white mensural ligatures.

To engrave white mensural ligatures, in the layout block, replace the \texttt{Ligature\_bracket\_engraver} with the \texttt{Mensural\_ligature\_engraver} in the Voice context:

\begin{verbatim}
\layout { 
\context { 
\Voice

\new MensuralVoice { 
\once \set suggestAccidentals = ##t 
bes4 a2 g2 \ficta fis8 \ficta e! fis2 g1 
} }
\end{verbatim}
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,

```latex
\score {
  \relative c' {
    \set Score.timing = ##f
    \set Score.defaultBarType = "-"
    \override NoteHead.style = #'petrucci
    \override Staff.TimeSignature.style = #'mensural
    \clef "petrucci-g"
    \[ c'\maxima g \]
    \[ d\longa \]
    \[ c'\maxima d\longa \]
    \[ e\breve a, g\breve \]
    \[ c'\maxima g \]
    \[ d\longa \]
    \[ c'\maxima d\longa \]
    \[ e\breve a, g\breve \]
  }
  \layout {
    \context {
      \Voice
      \remove "Ligature_bracket_engraver"
      \consists "Mensural_ligature_engraver"
    }
  }
}
```

Without replacing `Ligature_bracket_engraver` with `Mensural_ligature_engraver`, the same music transcribes to the following

```latex
\score {
  \relative c' {
    \set Score.timing = ##f
    \set Score.defaultBarType = "-"
    \override NoteHead.style = #'petrucci
    \override Staff.TimeSignature.style = #'mensural
    \clef "petrucci-g"
    \[ c'\maxima g \]
    \[ d\longa \]
    \[ c'\maxima d\longa \]
    \[ e\breve a, g\breve \]
    \[ c'\maxima g \]
    \[ d\longa \]
    \[ c'\maxima d\longa \]
    \[ e\breve a, g\breve \]
  }
  \layout {
    \context {
      \Voice
      \remove "Ligature_bracket_engraver"
      \consists "Mensural_ligature_engraver"
    }
  }
}
```
See also
Notation Reference: [Gregorian square neume ligatures], page 420, [Ligatures], page 407.

Known issues and warnings
Horizontal spacing of ligatures is 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.

See also
Notation Reference: [White mensural ligatures], page 414, [Ligatures], page 407.

Gregorian chant contexts
The predefined VaticanaVoiceContext and VaticanaStaffContext 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
    \[ f\melisma \pes a c' \pes d'\melismaEnd \] c' \divisioMinima \break
    \[ c'\melisma \flexa a \] \[ a \flexa \deminutum g\melismaEnd \] 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 \[Clef\], page 16. The note printed to the right side of each clef in the example column denotes the c' with respect to that clef.

<table>
<thead>
<tr>
<th>Description</th>
<th>Supported Clefs</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editio Vaticana style do clef</td>
<td>vaticana-do1, vaticana-do2, vaticana-do3</td>
<td></td>
</tr>
<tr>
<td>Editio Vaticana style fa clef</td>
<td>vaticana-fa1, vaticana-fa2</td>
<td></td>
</tr>
<tr>
<td>Editio Medicaea style do clef</td>
<td>medicaea-do1, medicaea-do2, medicaea-do3</td>
<td></td>
</tr>
<tr>
<td>Editio Medicaea style fa clef</td>
<td>medicaea-fa1, medicaea-fa2</td>
<td></td>
</tr>
<tr>
<td>hufnagel style do clef</td>
<td>hufnagel-do1, hufnagel-do2, hufnagel-do3</td>
<td></td>
</tr>
<tr>
<td>hufnagel style fa clef</td>
<td>hufnagel-fa1, hufnagel-fa2</td>
<td></td>
</tr>
<tr>
<td>hufnagel style combined do/fa clef</td>
<td>hufnagel-do-fa</td>
<td></td>
</tr>
</tbody>
</table>

See also

Notation Reference: \[Clef\], page 16.

Gregorian accidentals and key signatures

Accidental for the three different Gregorian styles are available:

\texttt{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

**See also**


Notation Reference: Section 1.1 [Pitches], page 1, [Accidentals], page 5, [Automatic accidentals], page 25, [Key signature], page 20.

Internals Reference: Section “KeySignature” in *Internals Reference*.

**Divisiones**

There are no rests in Gregorian chant notation; instead, it uses [Divisiones], page 418.

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], page 126. 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`

```
divisio minima  divisio maior  divisio maxima
+----------------+----------------+----------------+
|                |                |                |
|                |                |                |
|                |                |                |
|                |                |                |
|                |                |                |
|                |                |                |
|                |                |                |
|    finalis    |      virgula   |      caesura   |
+----------------+----------------+----------------+
|                |                |                |
|                |                |                |
|                |                |                |
|                |                |                |
|                |                |                |
|                |                |                |
```

**Predefined commands**

`\virgula, \caesura, \divisioMinima, \divisioMaior, \divisioMaxima, \finalis`.

**See also**


Notation Reference: [Breath marks], page 126.

Installed Files: ‘ly/gregorian.ly’.

**Gregorian articulation signs**

In addition to the standard articulation signs described in section [Articulations and ornamentations], page 111, 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 = #-.1
    \ictus_; ictus _|bar|_ break
    \circulus_; circulus _|bar|_ break
    \semicirculus_; semicirculus _|bar|_ break
    \accentus_; accentus _|bar|_ break
    \[ a_;episema \episemInitium \pes b \flexa a b \episemFinis \flexa a \]
  }
}

\text{ictus}
\text{circulus}
\text{semicirculus}
\text{accentus}
\text{episema}

See also
Notation Reference: [Articulations and ornamentations], page 111.
Snippets: Section “Ancient notation” in Snippets.

Known issues and warnings
Some articulations are vertically placed too closely to the corresponding note heads.

Augmentum dots (\textit{morae})
Augmentum dots, also called \textit{morae}, are added with the music function \texttt{\augmentum}. Note that \texttt{\augmentum} is implemented as a unary music function rather than as head prefix. It applies to the immediately following music expression only. That is, \texttt{\augmentum \virga c} will have no visible effect. Instead, say \texttt{\virga \augmentum c} or \texttt{\augmentum \{\virga c\}}. Also note that you can say \texttt{\augmentum \{a g\}} as a shortcut for \texttt{\augmentum a \augmentum g}. 
\include "gregorian.ly"
\score {
  \new VaticanaVoice {
    \[ \augmentum a \flexa \augmentum g \]
    \augmentum g
  }
}

See also
Notation Reference: [Breath marks], page 126.
Internals Reference: Section “BreathingSign” in Internals Reference.
Snippets: Section “Ancient notation” in Snippets.

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, \stropha, \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
Chapter 2: Specialist notation

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 \stropa.

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-notequilisma.

In addition to the note signs, ‘gregorian.ly’ also defines the commands \versus, \responsum, \ij, \iiij, \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.

<table>
<thead>
<tr>
<th>Basic and Liquescent forms</th>
<th>Output</th>
<th>LilyPond code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Punctum</td>
<td></td>
<td>[ b ]</td>
</tr>
</tbody>
</table>
\[ \cavum b \]

\[ \linea b \]

\[ \textit{Punctum Auctum Ascendens} \]
\[ \textit{Punctum Auctum Descendens} \]

\[ \textit{Punctum inclinatum} \]
\[ \textit{Punctum Inclinatum Auctum} \]
\[ \textit{Punctum Inclinatum Parvum} \]

\[ \textit{Virga} \]

\[ \textit{Two-note ligatures} \]
\[ \textit{Clivis vel Flexa} \]
\[ \textit{Clivis Aucta Descendens} \]
Chapter 2: Specialist notation

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
\[ a \pes b \flexa g \]

Chapter 2: Specialist notation

Torculus Auctus Descendens
\[ a \text{pes} b \text{flexa} \text{auctum} \text{descendens} g \]

Torculus Deminutus
\[ a \text{pes} b \text{flexa} \text{deminutum} g \]

Torculus Initio Debilis
\[ \text{deminutum} a \text{pes} b \text{flexa} g \]

Torculus Auctus Descendens Initio Debilis
\[ \text{deminutum} a \text{pes} b \text{flexa auctum} \text{descendens} g \]

Torculus Deminutus Initio Debilis
\[ \text{deminutum} a \text{pes} b \text{flexa deminutum} g \]

Porrectus
\[ a \text{flexa} g \text{pes} b \]

Porrectus Auctus Descendens
\[ a \text{flexa} g \text{pes} \text{auctum} \text{descendens} b \]

Porrectus Deminutus
\[ a \text{flexa} g \text{pes} \text{deminutum} b \]

Climacus
\[ \text{virga} b \text{inclinatum} a \text{inclinatum} g \]
Chapter 2: Specialist notation

\[ \text{Climacus Auctus} \]

\[ \text{\textbackslash virga b \textbackslash inclinatum a} \]
\[ \text{\textbackslash inclinatum \textbackslash auctum g} \]

\[ \text{Climacus Deminutus} \]

\[ \text{\textbackslash virga b \textbackslash inclinatum a} \]
\[ \text{\textbackslash inclinatum \textbackslash deminutum g} \]

\[ \text{Scandicus} \]

\[ \text{\textbackslash g \textbackslash pes a \textbackslash virga b} \]

\[ \text{Scandicus Auctus Descendens} \]

\[ \text{\textbackslash g \textbackslash pes a \textbackslash pes \textbackslash auctum} \]
\[ \text{\textbackslash descendens b} \]

\[ \text{Scandicus Deminutus} \]

\[ \text{\textbackslash g \textbackslash pes a \textbackslash pes \textbackslash deminutum b} \]

\[ \text{Special Signs} \]

\[ \text{Quilisma} \]

\[ \text{\textbackslash g \textbackslash pes \textbackslash quilisma a \textbackslash pes b} \]

\[ \text{Quilisma Pes Auctus Descendens} \]

\[ \text{\textbackslash quilisma g \textbackslash pes \textbackslash auctum} \]
\[ \text{\textbackslash descendens b} \]

\[ \text{Oriscus} \]

\[ \text{\textbackslash oriscus b} \]

\[ \text{Pes Quassus} \]

\[ \text{\textbackslash oriscus g \textbackslash pes \textbackslash virga b} \]
**Predefined commands**

The following head prefixes are supported: \texttt{\virga}, \texttt{\stropha}, \texttt{\inclinatum}, \texttt{\auctum}, \texttt{\descendens}, \texttt{\ascendens}, \texttt{\oriscus}, \texttt{\quilisma}, \texttt{\deminutum}, \texttt{\cavum}, \texttt{\linea}.

Head prefixes can be accumulated, though restrictions apply. For example, either \texttt{\descendens} or \texttt{\ascendens} can be applied to a head, but not both to the same head.

Two adjacent heads can be tied together with the \texttt{\pes} and \texttt{\flexa} infix commands for a rising and falling line of melody, respectively.

Use the unary music function \texttt{\augmentum} to add augmentum dots.
See also


Notation Reference: [Gregorian square neume ligatures], page 420, [White mensural ligatures], page 414, [Ligatures], page 407.

Known issues and warnings

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:

\score {
<<
\new KievanVoice = "melody" \relative c' {
\cadenzaOn
 c4 c c c c2 b\longa
\bar "k"
}
\new Lyrics \lyricsto "melody" {
 Го -- спо -- ди по -- мн -- луй.
}
>>
}

Господи помилуй.

See also


Known issues and warnings

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

\autoBeamOff\cadenzaOn\kievanOn b'1 b'2 b'4 b'8 b'\breve b'\longa\kievanOff b'2

\clef "kievan-do"
\kievanOn
\c

\autoBeamOff\cadenzaOn\kievanOn b'1 b'2 b'4 b'8 b'\breve b'\longa\kievanOff b'2

See also
Notation Reference: [Clef], page 16.

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

\clef "kievan-do"
\kievanOn
\c

\autoBeamOff\cadenzaOn\kievanOn
\c

\autoBeamOff\cadenzaOn\kievanOn b'1 b'2 b'4 b'8 b'\breve b'\longa\kievanOff b'2

See also
Notation Reference: Section A.9 [Note head styles], page 645.

Known issues and warnings
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,
\kievanOn
\clef "kievan-do"
c \bar "k"

See also
Notation Reference: [Accidentals], page 5, [Automatic accidentals], page 25, Section A.8 [The Feta font], page 624

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"

See also
Section 1.2.5 [Bars], page 90, Section A.8 [The Feta font], page 624

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 KievanLigature_engraver.

When the KievanVoice and KievanStaff contexts are used, the KievanLigature_engraver is enabled by default. In other contexts, it can be invoked by replacing the Ligature_bracket_engraver with the KievanLigature_engraver in the layout block:
\layout {
  \context {
    \Voice
    \remove "Ligature_bracket_engraver"
    \consists "KievanLigature_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:
\score {  
  \new KievanVoice = "melody" \relative c' {  
    \cadenzaOn  
    e2 [\ e4( d4 ) \] [\ c4( d e d ) \] e1 \bar "k"  
  }  
  \new Lyrics \lyricsto "melody" {  
    Га -- врі -- і -- лу  
  }  
}  

Га - врі - і - лу

See also


Notation Reference: [White mensural ligatures], page 414, [Gregorian square neume ligatures], page 420, [Ligatures], page 407.

Known issues and warnings

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

TBC

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 by setting the transparent property.

global = {  
  \hide Staff.BarLine
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:

```latex
\layout {
  ...
  \context {
    \Voice
      \remove "Stem_engraver"
  }
}
```

However, 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, and restore the stem when needed with the corresponding \once \override Stem.transparent = ##f (see example below).

**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:

```latex
\include "gregorian.ly"
chant = \relative c' {
  \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"
 \hide Stem
 }
}
}

\include "gregorian.ly"
\relative c' {
 \clef "G_8"
 \set Score.timing = ##f
 c\breve \hide NoteHead c c c c c
 \undo \hide NoteHead
 \override Stem.transparent = ##f \stemUp c4 b4 a
 \hide Stem c2 c4 \divisionMaioR
 c\breve \hide NoteHead 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.
}

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, here in combination with changing stem visibility:
\include "gregorian.ly"
\relative c' {
 \clef "G_8"
 \set Score.timing = ##f
 c\breve \hide NoteHead c c c c c
 \undo \hide NoteHead
 \override Stem.transparent = ##f \stemUp c4 b4 a
 \hide Stem c2 c4 \divisionMaioR
 c\breve \hide NoteHead 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.
}
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 a “notes”.

\score {
\new Staff <<
\new Voice = "melody" \chant
\new Lyrics \lyricsto "melody" \verba
>>
\layout {
\context {
\Staff
\remove "Time_signature_engraver"
\hide BarLine
\hide Stem
}
}
\override Lyrics.LyricText.X-extent = #'(0 . 3)
d4 \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 }

\score {
\new Staff <<
\new Voice = "chant" \spiritus
\new Lyrics = "one" \lyricsto "chant" \spirLyr
>>
\layout {
\context {

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 Section 1.1.1 [Writing pitches], page 1, and the various note names that may be used are explained in [Note names in other languages], page 7.

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 Section 2.10.2 [Arabic music], page 435.

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 Section 2.10.3 [Turkish classical music], page 440.

To locate init files such as ‘arabic.ly’ or ‘makam.ly’ on your system, see Section “Other sources of information” in Learning Manual.

Selected Snippets

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.keySignature = #`((6 . ,(- KOMA)) (3 . ,BAKIYE))
  c4 cc db fk
gbm4 gfc gfb efk
  fk4 db cc c
}

See also

Notation Reference: Section 1.1.1 [Writing pitches], page 1, [Note names in other languages], page 7, Section 2.10.2 [Arabic music], page 435, Section 2.10.3 [Turkish classical music], page 440.

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 Section 2.10.1 [Common notation for non-Western music], page 434.
• Additional key signatures can also be tailored as described in [Key signature], page 20.
• Complex time signatures may require that notes be grouped manually as described in [Manual beams], page 86.
• Takasim which are rhythmically free improvisations may be written down omitting bar lines as described in [Unmetered music], page 68.

See also
Notation Reference: Section 2.10.1 [Common notation for non-Western music], page 434, [Key signature], page 20, [Manual beams], page 86.
Snippets: Section “World music” in Snippets.

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 [Note names in other languages], page 7; the use of standard Western notation to notate non-Western music is discussed in Section 2.10.1 [Common notation for non-Western music], page 434.

For example, this is how the Arabic rast scale can be notated:
\include "arabic.ly"
\relative do' {
  do  re  misb  fa  sol  la  sisb  do  sisb  la  sol  fa  misb  re  do
}

\begin{music}
\include "arabic.ly"
\relative do' {
  \set Staff.extraNatural = ##f
dod dob dosd \down dob dobsb dodsd do do
}
\end{music}

The symbol for semi-flat does not match the symbol which is used in Arabic notation. The \down symbol defined in ‘arabic.ly’ may be used preceding a flat symbol as a work around if it is important to use the specific Arabic semi-flat symbol. The appearance of the semi-flat symbol in the key signature cannot be altered by using this method.

\include "arabic.ly"
\relative do' {
  \set Staff.extraNatural = ##f
  dod dob dosd \down dob dobsb dodsd do do
}

\begin{music}
\include "arabic.ly"
\relative do' {
  \set Staff.extraNatural = ##f
  dod dob dosd \down dob dobsb dodsd do do
}
\end{music}

See also
Notation Reference: [Note names in other languages], page 7, Section 2.10.1 [Common notation for non-Western music], page 434.
Snippets: Section “World music” in Snippets.
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:

<table>
<thead>
<tr>
<th>maqam group</th>
<th>key</th>
<th>finalis</th>
<th>Other maqmas in group (finalis)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ajam</td>
<td>major</td>
<td>sib</td>
<td>jaharka (fa)</td>
</tr>
<tr>
<td>bayati</td>
<td>bayati</td>
<td>re</td>
<td>hussaini, muhayer, saba, ushaq, nawa (sol)</td>
</tr>
<tr>
<td>hijaz</td>
<td>kurd</td>
<td>re</td>
<td>shahmaz, shad arban (sol), hijazkar (do)</td>
</tr>
<tr>
<td>iraq</td>
<td>iraq</td>
<td>sib</td>
<td>-</td>
</tr>
<tr>
<td>kurd</td>
<td>kurd</td>
<td>re</td>
<td>hijazkar kurd (do)</td>
</tr>
<tr>
<td>nahawand</td>
<td>minor</td>
<td>do</td>
<td>busalik (re), farah faza (sol)</td>
</tr>
<tr>
<td>nakriz</td>
<td>minor</td>
<td>do</td>
<td>nawa athar, hisar (re)</td>
</tr>
<tr>
<td>rast</td>
<td>rast</td>
<td>do</td>
<td>mahur, yakah (sol)</td>
</tr>
<tr>
<td>sikah</td>
<td>sikah</td>
<td>misb</td>
<td>huzam</td>
</tr>
</tbody>
</table>

Selected Snippets

Non-traditional key signatures

The commonly used \key command sets the keySignature 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.keySignature = #"(((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.) The accidentals in the key signature will appear in the reverse order to that in which they are specified.
Alternatively, for each item in the list, using the more concise format \texttt{(step . alter)} specifies that the same alteration should hold in all octaves.

For microtonal scales where a "sharp" is not 100 cents, \texttt{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 c' {
  \set Staff.keySignature = #`(((0 . 6) . ,FLAT)
  (((0 . 5) . ,FLAT)
  (((0 . 3) . ,SHARP))

c4 d e fis
aes4 bes c2
}
```

See also

Music Glossary: 
Section “maqam” in Music Glossary, 
Section “bayati” in Music Glossary, 
Section “rast” in Music Glossary, 
Section “sikah” in Music Glossary, 
Section “iraq” in Music Glossary, 
Section “kurd” in Music Glossary.

Notation Reference: [Key signature], page 20.


Internals Reference: Section “KeySignature” in Internals Reference.

Snippets: Section “World music” in Snippets, Section “Pitches” in Snippets.

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.

Selected Snippets

\textit{Arabic improvisation}

For improvisations or taqasim which are temporarily free, the time signature can be omitted and \texttt{\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
}
```
See also


Notation Reference: [Manual beams], page 86, [Automatic beams], page 76, [Unmetered music], page 68, [Automatic accidentals], page 25, [Setting automatic beam behavior], page 78, [Time signature], page 59.

Snippets: Section “World music” in Snippets.

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 re' {
    \set Staff.extraNatural = ##f
    \set Staff.autoBeaming = ##f
    \key re \bayati
    \time 10/8

    re4 re'8 re16 [misb re do] sisb [la sisb do] re4 r8
    do4 sisb8 misb16 [re do sisb] la [do sisb la] la4 r8
}
\header {
    title = "Semai Muhayer"
    composer = "Jamil Bek"
}
}
```

See also

Snippets: Section “World music” in Snippets.
Further reading for Arabic music


   There are also various web sites that explain maqams and some provide audio examples such as:
   - [http://www.turath.org/](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 maqam.

   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 Section 2.10.1 [Common notation for non-Western music], page 434.

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), *kücük mücenneb* (5/9), *büyük mücenneb* (8/9), *tanını* (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.

<table>
<thead>
<tr>
<th>Accidental name</th>
<th>suffix</th>
<th>pitch alteration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>bm</td>
<td>büyük mücenneb (sharp)</td>
<td>+8/9</td>
</tr>
<tr>
<td>k</td>
<td>kiçük mücenneb (sharp)</td>
<td>+5/9</td>
</tr>
<tr>
<td>-b</td>
<td>bakiye (sharp)</td>
<td>+4/9</td>
</tr>
<tr>
<td>-c</td>
<td>koma (sharp)</td>
<td>+1/9</td>
</tr>
<tr>
<td>fc</td>
<td>koma (flat)</td>
<td>-1/9</td>
</tr>
<tr>
<td>fb</td>
<td>bakiye (flat)</td>
<td>-4/9</td>
</tr>
<tr>
<td>fk</td>
<td>kiçük mücenneb (flat)</td>
<td>-5/9</td>
</tr>
<tr>
<td>fbm</td>
<td>büyük mücenneb (flat)</td>
<td>-8/9</td>
</tr>
</tbody>
</table>

For a more general explanation of non-Western music notation, see Section 2.10.1 [Common notation for non-Western music], page 434.

**See also**


Notation Reference: Section 2.10.1 [Common notation for non-Western music], page 434.
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 {
  ...
}

**Note:** 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' }

\[\begin{align*}
\text{\new Staff \{ c'4 c' c' c' \}} \\
\text{\new Staff \{ d'4 d' d' d' \}}
\end{align*}\]

\[\begin{align*}
\text{\new GrandStaff \textless~\textgreater~} \\
\text{\new StaffGroup \textless~\textgreater~} \\
\text{\new Staff \{ \flute \}} \\
\text{\new Staff \{ \oboe \}}
\end{align*}\]

\[\begin{align*}
\text{\new StaffGroup \textless~\textgreater~} \\
\text{\new Staff \{ \violinI \}} \\
\text{\new Staff \{ \violinII \}}
\end{align*}\]
Comments are one exception to this general rule. (For others see Section 3.1.5 [File structure], page 446.) 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 Section 3.1.2 [Multiple scores in a book], page 443, Section 3.1.3 [Multiple output files from one input file], page 444, Section 3.1.5 [File structure], page 446.)

See also


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"
}
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 Section 3.1.5 [File structure], page 446.

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 Section 4.2.1 [The \layout block], page 504.

• 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], page 448).

• 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 { }
}
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.

See also


Notation Reference: [Titles explained], page 448, Section 4.2.1 [The `\layout` block], page 504.
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 Section 3.1.5 [File structure], page 446. 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], page 451). 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.

Note: 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 g, {
    \clef bass
    \key g \major
    \repeat unfold 2 { g16( 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 b {
    \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.

\[ Image of music notation for Prélude. \]

Allemande.

\[ Image of music notation for Allemande. \]

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 tagline 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 VOCI"
            opus = "BWV 846"
            % Do not display the subtitle for this score
            subtitle = ##f
        }
    }
}
DAS WOHLTEMPERIRTE CLAVIER
TEIL I

PRAELUDIUM I

FUGA I
A 4 VOCI

See also
Notation Reference: Section 3.1.5 [File structure], page 446, [Default layout of bookpart and score titles], page 451, [Custom layout for titles], page 456.

Default layout of bookpart and score titles
This example demonstrates all \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 = "tagline goes at the bottom of the last page"
copyright = "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
Piece 1
Opus 1
\timesig.C44/clefs.G

Piece 2 on the same page
Opus 2
\timesig.C44/clefs.G

copyright goes at the bottom of the first page
Chapter 3: General input and output

Note that

- The instrument name will be repeated on every page.
- Only piece and opus are printed in a \score block 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], page 456.

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.

```latex
\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}
```
See also

Learning Manual: Section “How LilyPond input files work” in Learning Manual,
Notation Reference: [Custom layout for titles], page 456, Section 3.1.5 [File structure],
page 446.
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:
Chapter 3: General input and output

- page numbers are automatically placed on the top far left (if even) or top far right (if odd), starting from the second page.
- the \texttt{instrument} text field is placed in the center of every page, starting from the second page.
- the \texttt{copyright} text is centered on the bottom of the first page.
- the \texttt{tagline} is centered on the bottom of the last page, and below the \texttt{copyright} text if there is only a single page.

The default tagline can be changed by adding a \texttt{tagline} in the top-level \header block.

\book {
  \header {
      tagline = "... music notation for Everyone"
  }
  \score {
    \relative c' {
      c4 d e f
    }
  }
}

\begin{figure}
  \centering
  \includegraphics[width=\textwidth]{example.png}
  \caption{Example figure}
  \end{figure}

... music notation for Everyone

To remove the \texttt{tagline} set the value to \texttt{##f}.

3.2.2 Custom titles, headers and footers

Custom text formatting for titles

Standard \texttt{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" }
  }
}

\textbf{PRAELUDIUM I} \quad BWV 846
See also
Notation Reference: Section 1.8.2 [Formatting text], page 223.

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], page 451.

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 } } }\header { tagline = ##f }

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 } } } \header { tagline = ##f } }
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], page 448.

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"
    }
  }
}
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
}
}
\fromproperty #'header:opus
}
}
\header { tagline = ##f }
\score {
{ s1 }
\header {
piece = "FUGA I"
mycustomtext = "A 4 VOCI" %% User-defined field
opus = "BWV 846"
}
}
}

See also
Notation Reference: [Titles explained], page 448.
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:

\begin{verbatim}
variable = \markup {
... \\
\on-the-fly \procedure markup \\
...
}
\end{verbatim}

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:

<table>
<thead>
<tr>
<th>Procedure name</th>
<th>Condition tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>print-page-number-check-first</td>
<td>should this page number be printed?</td>
</tr>
<tr>
<td>create-page-number-stencil</td>
<td>print-page-numbers true?</td>
</tr>
<tr>
<td>print-all-headers</td>
<td>print-all-headers true?</td>
</tr>
<tr>
<td>first-page</td>
<td>first page in the book?</td>
</tr>
<tr>
<td>(on-page nmbr)</td>
<td>page number = nmbr?</td>
</tr>
<tr>
<td>last-page</td>
<td>last page in the book?</td>
</tr>
<tr>
<td>not-first-page</td>
<td>not first page in the book?</td>
</tr>
<tr>
<td>part-first-page</td>
<td>first page in the book part?</td>
</tr>
<tr>
<td>part-last-page</td>
<td>last page in the book part?</td>
</tr>
<tr>
<td>not-single-page</td>
<td>pages in book part &gt; 1?</td>
</tr>
</tbody>
</table>

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:

\begin{verbatim}
\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
      }
    }
  }
}
\end{verbatim}
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.

See also

Notation Reference: [Titles explained], page 448, [Default layout of bookpart and score titles], page 451.

Installed Files: ‘..//ly/titling-init.ly’.

3.2.3 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:
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:

direction  If (and only if) the \footnote is being applied to a post-event or articulation, it must be preceded with a direction indicator (–, _, ^) in order to attach music (with a footnote mark) to the preceding note or rest.

mark  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 \default) 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  
  }  
}
Chapter 3: General input and output

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
  }
}

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 c'' {
    a4_\footnote #'(0 . -1) "A slur forced down" ( 
    b8^\footnote #'(1 . 0.5) "A manual beam forced up" [ 
    b8 ]
    c4 )
    c-\footnote #'(1 . 1) "Tenuto" --
  }
}
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
  }
}

\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
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:
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(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

**Note:** 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
  }
}

\[ \begin{array}{cccc}
\text{1} & \text{2} & \text{3} & \text{4} \\
\text{Meter change} & \text{Chord stem} & \text{Bar line} & \text{Key change} \\
\end{array} \]

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
  }
}
More examples of custom marks are shown in [Footnotes in stand-alone text], page 466.

**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

```latex
\markup { ... \auto-footnote \text \footnote \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:

```latex
\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 c' {
    a'4 b8 e c4 d
  }
}
```
A simple tune\footnote{By me} is shown below. It is a recent\footnote{Aug 2012} composition.

\begin{music}
\relative c'  {
  \n  a'4  b8  e  c4  d 
}
\end{music}

---

\footnotes\footnote{By me} \footnote{Aug 2012}

Footnotes in stand-alone text with custom marks

The syntax of a footnote in stand-alone text with custom marks is

\begin{verbatim}
\markup { ... \footnote \mark \footnote ... }
\end{verbatim}

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], page 460. Alternatively, ASCII aliases may be used (see [ASCII aliases], page 480):

\begin{verbatim}
\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 c' {
    a'4 b8 e c4 d
  }
}
\end{verbatim}
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], page 479):

```
\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 c' {
    a'4 b8 e c4 d
  }
}
```

A simple tune § is shown below. It is a recent ¶ composition.

---

§ By me
¶ Aug 2012

See also

Notation Reference: [ASCII aliases], page 480, [Balloon help], page 211, Section A.12 [List of special characters], page 697, [Text marks], page 219, [Text scripts], page 216, [Unicode], page 479.

Internals Reference: Section “FootnoteEvent” in Internals Reference, Section “FootnoteItem” in Internals Reference, Section “FootnoteSpanner” in Internals Reference, Section “Footnote_engraver” in Internals Reference.

Known issues and warnings
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.

3.2.4 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 {
  \c1
  \pageBreak \mark A \label #'markA
  \c1
  }
  \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", i.e. a two digit number.

**Predefined commands**

\label, \page-ref.

### 3.2.5 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 % ...
}

The markups which are used to format the table of contents are defined in the `\paper` block. The default ones are `tocTitleMarkup`, for formatting the title of the table, and `tocItemMarkup`, for formatting the toc elements, composed of the element title and page number. These variables may be changed by the user:

\paper {
  \% Translate the toc title into French:
  tocTitleMarkup = \markup \huge \column {
    \fill-line { \null "Table des matières" \null }
    \hspace #1}
  \% use larger font size
  tocItemMarkup = \markup \large \fill-line {
    \fromproperty #'toc:text \fromproperty #'toc:page}
}

Note how the toc element text and page number are referred to in the `tocItemMarkup` definition.
New commands and markups may also be defined to build more elaborated table of contents:

- first, define a new markup variable in the \paper block
- then, define a music function which aims at adding a toc element using this markup paper variable.

In the following example, a new style is defined for entering act names in the table of contents of an opera:

\paper {
  tocActMarkup = \markup \large \column {
    \hspace #1
    \fill-line { \null \italic \fromproperty #'toc:text \null }
    \hspace #1
  }
}
tocAct =
#(define-music-function (parser location text) (markup?)
  (add-toc-item! 'tocActMarkup text))

Table of Contents

Atto Primo
  Coro. Viva il nostro Alcide
  Cesare. Presti omai l’Egizzia terra

Atto Secondo
  Sinfonia
  Cleopatra. V’adoro, pupille, saette d’Amore

Dots can be added to fill the line between an item and its page number:
\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
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 Section 3.3.2 [Different editions from one source], page 473.

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
  ...

See also

Installed Files: ‘ly/toc-init.ly’.
then the entry file, `libA.ly`, will contain

```latex
#(ly:set-option 'relative-includes #t)
\include "A1.ly"
\include "A2.ly"
...
\%
```

return to default setting
```latex
#(ly:set-option 'relative-includes #f)
```

Any `.ly` file can then include the entire library simply with
```latex
\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
```bash
lilypond --include=parts main.ly
```

and in main.ly write
```latex
\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 Section “Other sources
of information” in Learning Manual). 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 Section
“Other sources of information” in Learning Manual. 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 Section “Scores and parts” in Learning
Manual.

See also

Learning Manual: Section “Other sources of information” in Learning Manual, Section
“Scores and parts” in Learning Manual.

Known issues and warnings

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 Section “Organizing pieces with variables” in Learning Manual. 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], page 167. Here is an example:

```latex
sopranoMusic = \relative c'' { a4 b c b8( a) }
sopranoMusic = \relative g' { e4 e e f }
tenorMusic = \relative c' { c4 b e d8( c) }
bassMusic = \relative c' { 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" {
    \set Staff.printPartCombineTexts = ##f
    \partcombine \sopranoMusic \altoMusic
  }
  \new Staff = "LH" {
    \set Staff.printPartCombineTexts = ##f
    \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 Section 3.3.1 [Including LilyPond files], page 472.

Using tags

The \( \texttt{\textbackslash tag \#'partA} \) command marks a music expression with the name \textit{partA}. Expressions tagged in this way can be selected or filtered out by name later, using either \texttt{\textbackslash keepWithTag \#'name} or \texttt{\textbackslash removeWithTag \#'name}. The result of applying these filters to tagged music is as follows:

<table>
<thead>
<tr>
<th>Filter</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tagged music preceded by \texttt{\textbackslash keepWithTag #'name} or \texttt{\textbackslash keepWithTag #'(name1 name2...)}</td>
<td>Untagged music and music tagged with any of the given tag names is included; music tagged with any other tag name is excluded.</td>
</tr>
<tr>
<td>Tagged music preceded by \texttt{\textbackslash removeWithTag #'name} or \texttt{\textbackslash removeWithTag #'(name1 name2...)}</td>
<td>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.</td>
</tr>
<tr>
<td>Tagged music not preceded by either \texttt{\textbackslash keepWithTag} or \texttt{\textbackslash removeWithTag}</td>
<td>All tagged and untagged music is included.</td>
</tr>
</tbody>
</table>

The arguments of the \texttt{\textbackslash tag}, \texttt{\textbackslash keepWithTag} and \texttt{\textbackslash removeWithTag} commands should be a symbol (such as \texttt{\#'score} or \texttt{\#'part}), followed by a music expression.

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' {
  g8. c32 d
  \tag \#'trills { d8.\trill }
  \tag \#'expand { \repeat unfold 3 { e32 d } }
  c32 d
}
```
Alternatively, it is sometimes easier to exclude sections of music:

\begin{verbatim}
music = \relative g' { 
  g8. c32 d
  \tag #'trills { d8.\trill }
  \tag #'expand {\repeat unfold 3 { e32 d } }
  c32 d
}
\end{verbatim}

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:

\begin{verbatim}
\tag #'finger ^4
\tag #'warn ^"Watch!"
\end{verbatim}

Multiple tags may be placed on expressions with multiple \tag entries, or by combining multiple tags into one symbol list:
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  
}
```

Two or more `\keepWithTag` filters applied to a single music expression will cause all tagged sections to be removed, as the first filter will remove all tagged sections except the one named, and the second filter will remove even that tagged section. Usually you would rather want to use a single `\keepWithTag` command with a list of multiple tags: this will only remove tagged sections not given in any of the tags.

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:
test = { \tag #'here \{ \tag #'here <<c'>> \} }

{\pushToTag #'here c' \pushToTag #'here e' \pushToTag #'here g' \test \appendToTag #'here c' \appendToTag #'here e' \appendToTag #'here g' \test }

Both commands get a tag, the material to splice in at every occurrence of the tag, and the
tagged expression. The commands make sure to copy everything that they change so that the
original \test retains its meaning.

See also


Notation Reference: [Automatic part combining], page 167, Section 3.3.1 [Including LilyPond files], page 472.

Known issues and warnings

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 -dinclude-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 Section “Style
sheets” in Learning Manual.

See also

Learning Manual: Section “Organizing pieces with variables” in Learning Manual, Section
“Style sheets” in Learning Manual.

Notation Reference: Section 3.3.1 [Including LilyPond files], page 472.

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:

```
\score {  
  \relative c'' { 
    c1 \mark \markup { \char ##x03EE } 
    c1_ \markup { \tiny { \char ##x03B1 " to " \char ##x03C9 } } 
  } 
  \addlyrics { O \markup { \concat { Ph \char ##x0153 be! } } } 
}
```

**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/](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'' { 
    c1 \mark \markup { \char ##x03EE } 
    c1_ \markup { \tiny { \char ##x03B1 " to " \char ##x03C9 } } 
  } 
  \addlyrics { O \markup { \concat { Ph \char ##x0153 be! } } } 
}
```
Copyright 2008--2012 ©

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; – &OE;uvre incompl` ete… &frqq;"
\score {
  \new Staff { \repeat unfold 9 a'4 }
  \addlyrics {
    This is al -- so wor -- kin’~in ly -- rics: &ndash;_&OE;…
  }
}
\markup \column {
  "The replacement can be disabled:"
  "&ndash; &OE; &hellip;"
  \override #'(replacement-alist . ()) "&ndash; &OE; &hellip;"
}

« – Œuvre incomplète… »

This is al-so workin’in lyrics: – Œ…

The replacement can be disabled:
– Œ…
&ndash; &OE; &hellip;

You can also make your own aliases, either globally:
Chapter 3: General input and output

\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.

See also
Notation Reference: Section A.12 [List of special characters], page 697.

3.4 Controlling output

3.4.1 Extracting fragments of music
It is possible to quote small fragments of a large score directly from the output. This can be
compared to clipping a piece of a paper score with scissors.

This is done by defining the measures that need to be cut out separately. For example,
including the following definition

\layout {
  clip-regions
  = #(list
    (cons
      (make-rhythmic-location 5 1 2)
      (make-rhythmic-location 7 3 4)))
}

will extract a fragment starting halfway the fifth measure, ending in the seventh measure. The
meaning of 5 1 2 is: after a 1/2 note in measure 5, and 7 3 4 after 3 quarter notes in measure 7.

More clip regions can be defined by adding more pairs of rhythmic-locations to the list.

In order to use this feature, LilyPond must be invoked with ‘-dclip-systems’. The clips
are output as EPS files, and are converted to PDF and PNG if these formats are switched on
as well.

For more information on output formats, see Section “Invoking lilypond” in Application
Usage.

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
{ ... } 

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.

c8 d
\set Score.skipTypesetting = ##t
e8 e e e e e e
\set Score.skipTypesetting = ##f
c8 d b bes a g c2

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). Scalable Vector Graphics (SVG), Encapsulated PostScript (EPS) and Portable Network Graphics (PNG) output formats are also available through command line options, see Section “Basic command line options for LilyPond” in Application Usage.

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/

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:

![Sample bars of music set in LilyPond’s Feta font](image)

### Installation Instructions for MacOS

Download and extract the zip file. Copy the `lilyfonts` directory to `'SHARE_DIR/lilypond/current`; for more information, see Section “Other sources of information” in Learning Manual. Rename the existing `fonts` directory to `fonts_orig` and the `lilyfonts` directory to `fonts`. To revert back to Feta, reverse the process.

See also


### Known issues and warnings

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 MIDI output

MIDI (Musical Instrument Digital Interface) is a standard for connecting and controlling digital instruments. A MIDI file is a series of notes in a number of tracks. It is not an actual sound file; you need special software to translate between the series of notes and actual sounds.

Pieces of music can be converted to MIDI files, so you can listen to what was entered. This is convenient for checking the music; octaves that are off or accidentals that were mistyped stand out very much when listening to the MIDI output.

Standard MIDI output is somewhat crude; optionally, an enhanced and more realistic MIDI output is available by means of Section 3.5.7 [The Articulate script], page 492.

The MIDI output allocates a channel for each staff, and reserves channel 10 for drums. There are only 16 MIDI channels per device, so if the score contains more than 15 staves, MIDI channels will be reused.

#### 3.5.1 Creating MIDI files

To create a MIDI output file from a LilyPond file, insert a `\midi` block inside a `\score` block:

```latex
\score {
  ...music...
  \layout { }
  \midi { }
}
```
If there is only a `\midi` block in a `\score` (i.e. without any `\layout` block), then only MIDI output will be produced. No musical notation will be typeset.

\score {
  ...music...
  \midi { }
}

Dynamics, pitches, rhythms, tempo changes and ties are all interpreted and translated correctly. Dynamic ‘marks’ translate into volume levels with a ‘fixed fraction’ of the available MIDI volume range; crescendi and decrescendi make the volume vary linearly between their two extremes.

All `\tempo` indications, including all subsequent tempo changes, specified within the music notation will be reflected in the MIDI output.

Usually it is enough to leave the `\midi` block empty, but it can contain context rearrangements, new context definitions or code to set the values of properties. Here the tempo is set to 72 quarter-note beats per minute, but only for the MIDI’s audio playback.

\score {
  ...music...
  \midi {
    \tempo 4 = 72
  }
}

Note that `\tempo` is actually a command for setting properties during the interpretation of the music and in the context of output definitions, like a `\midi` block, is reinterpreted 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"
    }
  }
}

removes the effect of dynamics from the MIDI output. Translation modules for sound are called ‘performers’.

**Selected Snippets**

*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 MIDI port, and most devices support only one port.

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
}

\begin{music}
\begin{musicnote}[\score]{Flute}
\begin{musicnote}[\score]{Clarinet}
\end{musicnote}
\end{music}

Known issues and warnings

Some operating systems require a specific file extension for MIDI files. If a different extension is preferred insert the following line at the top-level of the input file, before the start of any \book, \bookpart or \score blocks:

\%(ly:set-option 'midi-extension "mid")

This will set the default extension for MIDI files to .mid.

Alternatively, an option can be supplied on the command line:

lilypond -dmidi-extension=mid MyFile.ly

Changes in the MIDI volume take place only on starting a note, so crescendi and decrescendi cannot affect the volume of a single note.

Some MIDI players may not always correctly handle tempo changes in the midi output.
See also

Installed Files: ‘../ly/performer-init.ly’.

3.5.2 MIDI Instruments

The MIDI instrument to be used is specified by setting the \texttt{Staff.midiInstrument} property to the instrument name. The name should be chosen from the list in Section A.6 [MIDI instruments], page 622.

\begin{verbatim}
\new Staff {
  \set Staff.midiInstrument = #"glockenspiel"
  ...notes...
}
\new Staff \with {midiInstrument = #"cello"} {
  ...notes...
}
\end{verbatim}

If the selected instrument does not exactly match an instrument from the list of MIDI instruments, the Grand Piano ("acoustic grand") instrument is used.

3.5.3 What goes into the MIDI output?

Supported in MIDI

The following items of notation are reflected in the MIDI output:

- Pitches
- Microtones (See [Accidentals], page 5. Rendering needs a player that supports pitch bend.)
- Chords entered as chord names
- Rhythms entered as note durations, including tuplets
- Tremolos entered without ‘:[number]’
- Ties
- Dynamic marks
- Crescendi, decrescendi over multiple notes
- Tempo changes entered with a tempo marking
- Lyrics

Using Section 3.5.7 [The Articulate script], page 492, a number of items are added to the above list:

- Articulations (slurs, staccato, etc)
- Trills, turns
- Rallentando and accelerando

Unsupported in MIDI

The following items of notation have no effect on the MIDI output, unless you use Section 3.5.7 [The Articulate script], page 492:

- Rhythms entered as annotations, e.g. swing
- Tempo changes entered as annotations with no tempo marking
- Staccato and other articulations and ornamentations
- Slurs and Phrasing slurs
- Crescendi, decrescendi over a single note
• Tremolos entered with ‘:[number]’
• Figured bass
• Microtonal chords

3.5.4 Repeats in MIDI

With a few minor additions, all types of repeats can be represented in the MIDI output. This is achieved by applying the `\unfoldRepeats` music function. This function changes all repeats to unfold repeats.

```plaintext
\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' }
  }
}
\bar "|."
```

In scores containing multiple voices, unfolding of repeats in MIDI output will only occur correctly if each voice contains fully notated repeat indications.

When creating a score file using `\unfoldRepeats` for MIDI, it is necessary to make two `\score` blocks: one for MIDI (with unfolded repeats) and one for notation (with volta, tremolo, and percent repeats). For example,

```plaintext
\score {
  ...music...
  \layout { ... }
}
\score {
  \unfoldRepeats ...music...
  \midi { ... }
}
```

3.5.5 Controlling MIDI dynamics

MIDI dynamics are implemented by the Dynamic performer which lives by default in the Voice context. It is possible to control the overall MIDI volume, the relative volume of dynamic markings and the relative volume of different instruments.
Dynamic marks

Dynamic marks are translated to a fixed fraction of the available MIDI volume range. The default fractions range from 0.25 for \textit{ppppp} to 0.95 for \textit{fffff}. The set of dynamic marks and the associated fractions can be seen in ‘../scm/midi.scm’, see Section “Other sources of information” in \textit{Learning Manual}. This set of fractions may be changed or extended by providing a function which takes a dynamic mark as its argument and returns the required fraction, and setting \texttt{Score.dynamicAbsoluteVolumeFunction} to this function.

For example, if a \textit{rinforzando} dynamic marking, \texttt{rfz}, is required, this will not by default have any effect on the MIDI volume, as this dynamic marking is not included in the default set. Similarly, if a new dynamic marking has been defined with \texttt{make-dynamic-script} that too will not be included in the default set. The following example shows how the MIDI volume for such dynamic markings might be added. The Scheme function sets the fraction to 0.9 if a dynamic mark of \texttt{rfz} is found, or calls the default function otherwise.

\begin{verbatim}
#(define (myDynamics dynamic)
  (if (equal? dynamic "rfz")
    0.9
    (default-dynamic-absolute-volume dynamic)))
\end{verbatim}

```
\score {
  \new Staff {
    \set Staff.midiInstrument = \"cello\"
    \set Score.dynamicAbsoluteVolumeFunction = #myDynamics
    \new Voice {
      \relative c'' {
        a4\pp b c-\rfz
      }
    }
  }
  \layout {}
  \midi {}
}
```

Alternatively, if the whole table of fractions needs to be redefined, it would be better to use the \texttt{default-dynamic-absolute-volume} procedure in ‘../scm/midi.scm’ and the associated table as a model. The final example in this section shows how this might be done.

Overall MIDI volume

The minimum and maximum overall volume of MIDI dynamic markings is controlled by setting the properties \texttt{midiMinimumVolume} and \texttt{midiMaximumVolume} at the \texttt{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

\[
\text{midiMinimumVolume} + (\text{midiMaximumVolume} - \text{midiMinimumVolume}) \times \text{fraction}
\]

In the following example the dynamic range of the overall MIDI volume is limited to the range 0.2 - 0.5.

```
\score {
  \<<
```

---

\texttt{rfz}
Equalizing different instruments (i)

If the minimum and maximum MIDI volume properties are set in the `Staff` context the relative volumes of the MIDI instruments can be controlled. This gives a basic instrument equalizer, which can enhance the quality of the MIDI output remarkably.

In this example the volume of the clarinet is reduced relative to the volume of the flute.

```ansi
\score {
<<
  \new Staff {
    \key g \major
    \time 2/2
    \set Staff.midiInstrument = #"flute"
    \new Voice \relative c''' {
      r2 g\mp g fis~
      fis4 g8 fis e2~
      e4 d8 cis d2
    }
  }
  \new Staff {
    \key g \major
    \set Staff.midiInstrument = #"clarinet"
    \new Voice \relative c'' {
      b1\p a2. b8 a
      g2. fis8 e
      fis2 r
    }
  }
  >>
\layout {}
\midi {
  \tempo 2 = 72
  \context {
    \Score
      midiMinimumVolume = #0.2
      midiMaximumVolume = #0.5
  }
}
}
```
Equalizing different instruments (ii)

If the MIDI minimum and maximum volume properties are not set LilyPond will, by default, apply a small degree of equalization to a few instruments. The instruments and the equalization applied are shown in the table `instrument-equalizer-alist` in `../scm/midi.scm`.

This basic default equalizer can be replaced by setting `instrumentEqualizer` in the `Score` context to a new Scheme procedure which accepts a MIDI instrument name as its only argument and returns a pair of fractions giving the minimum and maximum volumes to be applied to that instrument. This replacement is done in the same way as shown for resetting the `dynamicAbsoluteVolumeFunction` at the start of this section. The default equalizer, `default-instrument-equalizer`, in `../scm/midi.scm` shows how such a procedure might be written.

The following example sets the relative flute and clarinet volumes to the same values as the previous example.

```scheme
#(define my-instrument-equalizer-alist '())

#(set! my-instrument-equalizer-alist
   (append
    '(
      ("flute" . (0.7 . 0.9)))
```
Chapter 3: General input and output

3.5.6 Percussion in MIDI

Percussion instruments are generally notated in a DrumStaff context and when notated in this way they are outputted correctly to MIDI channel 10, but some pitched percussion instruments, like the xylophone, marimba, vibraphone, timpani, etc., are treated like “normal” instruments and music for these instruments should be entered in a normal Staff context, not a DrumStaff context, to obtain the correct MIDI output.
Some non-pitched percussion sounds included in the general MIDI standard, like melodic tom, taiko drum, synth drum, etc., cannot be reached via MIDI channel 10, so the notation for such instruments should also be entered in a normal Staff context, using suitable normal pitches.

Many percussion instruments are not included in the general MIDI standard, e.g. castanets. The easiest, although unsatisfactory, method of producing some MIDI output when writing for such instruments is to substitute the nearest sound from the standard set.

**Known issues and warnings**

Because the general MIDI standard does not contain rim shots, the sidestick is used for this purpose instead.

### 3.5.7 The Articulate script

A more realistic MIDI output is possible when using the Articulate script. It tries to take articulations (slurs, staccato, etc) into account, by replacing notes with sequential music of suitably time-scaled note plus skip. It also tries to unfold trills turns etc., and take rallentando and accelerando into account.

To use the Articulate script, you have to include it at the top of your input file,

```
\include "articulate.ly"
```

and in the `\score` section do

```
\unfoldRepeats \articulate <<
all the rest of the score...
>>
```

After altering your input file this way, the visual output is heavily altered, but the standard `\midi` block will produce a better MIDI file.

Although not essential for the Articulate script to work, you may want to insert the `\unfoldRepeats` command as it appears in the example shown above as it enables performing abbreviations such as *trills*.

**Known issues and warnings**

Articulate shortens chords and some music (esp. organ music) could sound worse.

### 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 cis e fis g }
```

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). This is convenient since you can just insert \displayLilyMusic into existing music in order to get information about it. If you don’t actually want Lilypond to interpret the displayed music as well as display it, use \void in order to have it ignored:

\begin{verbatim}
\void \displayLilyMusic \transpose c \, a, \{ \, c4 \, e \, g \, a \, bes \}
\end{verbatim}

3.6.2 Displaying scheme music expressions

See Section “Displaying music expressions” in Extending.

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:

\begin{verbatim}
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
\end{verbatim}

The syntax is a tab-delimited line, with two fixed fields on each line followed by optional parameters.

\verb|time| \, \verb|type| \, ...\verb|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.

Known issues and warnings

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: \texttt{paper \{\ldots\}} and \texttt{layout \{\ldots\}}. The \texttt{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 Section 4.1 [Page layout], page 494. The \texttt{layout} block contains score layout settings, such as the number of systems to use, or the space between staff-groups, etc. See Section 4.2 [Score layout], page 504.

4.1 Page layout

This section discusses page layout options for the \texttt{paper} block.

4.1.1 The \texttt{paper} block

\texttt{paper} blocks may be placed in three different places to form a descending hierarchy of \texttt{paper} blocks:

- At the top of the input file, before all \texttt{book}, \texttt{bookpart}, and \texttt{score} blocks.
- Within a \texttt{book} block but outside all the \texttt{bookpart} and \texttt{score} blocks within that book.
- Within a \texttt{bookpart} block but outside all \texttt{score} blocks within that bookpart.

A \texttt{paper} block cannot be placed within a \texttt{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 \texttt{paper} blocks can appear at each of the levels, for example as parts of several \texttt{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 \texttt{paper} block include:

- the \texttt{set-paper-size} scheme function,
- \texttt{paper} variables used for customizing page layout, and
- markup definitions used for customizing the layout of headers, footers, and titles.

The \texttt{set-paper-size} function is discussed in the next section, Section 4.1.2 [Paper size and automatic scaling], page 495. The \texttt{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 Section 3.2.2 [Custom titles headers and footers], page 455.

Most \texttt{paper} variables will only work in a \texttt{paper} block. The few that will also work in a \texttt{layout} block are listed in Section 4.2.1 [The \texttt{layout} block], page 504.

Except when specified otherwise, all \texttt{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 \texttt{top-margin} to ten millimeters:
To set it to 0.5 inches, use the \texttt{\textbackslash in} unit suffix:

\begin{verbatim}
\texttt{paper \{ 
\hspace{1em}top-margin = 0.5\texttt{\textbackslash in} \}}
\end{verbatim}

The available unit suffixes are \texttt{\textbackslash mm}, \texttt{\textbackslash cm}, \texttt{\textbackslash in}, and \texttt{\textbackslash pt}. These units are simple values for converting from millimeters; they are defined in \texttt{ly/paper-defaults-init.ly}. For the sake of clarity, when using millimeters, the \texttt{\textbackslash mm} is typically included in the code, even though it is not technically necessary.

It is also possible to define \texttt{paper} values using Scheme. The Scheme equivalent of the above example is:

\begin{verbatim}
\texttt{paper \{ 
\hspace{1em}#(define top-margin (* 0.5 in)) \}}
\end{verbatim}

\textbf{See also}

Notation Reference: Section \texttt{4.1.2 [Paper size and automatic scaling]}, page 495, Section \texttt{3.2.2 [Custom titles headers and footers]}, page 455, Section \texttt{4.2.1 [The \texttt{\textbackslash layout} block]}, page 504.

Installed Files: \texttt{`ly/paper-defaults-init.ly’}.

\subsection*{4.1.2 Paper size and automatic scaling}

\textbf{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:

\begin{verbatim}
\texttt{set-default-paper-size 
\hspace{1em}#(set-default-paper-size "quarto")}
\end{verbatim}

which must always be placed at the toplevel scope, and

\begin{verbatim}
\texttt{set-paper-size 
\hspace{1em}\texttt{paper \{ 
\hspace{2em}#(set-paper-size "tabloid") \}}}
\end{verbatim}

which must always be placed in a \texttt{paper} block.

If the \texttt{set-default-paper-size} function is used in the toplevel scope, it must come before any \texttt{paper} block. \texttt{set-default-paper-size} sets the paper size for all pages, whereas \texttt{set-paper-size} only sets the paper size for the pages that the \texttt{paper} block applies to. For example, if the \texttt{paper} block is at the top of the file, then it will apply the paper size to all pages. If the \texttt{paper} block is inside a \texttt{book}, then the paper size will only apply to that book.

When the \texttt{set-paper-size} function is used, it must be placed \textit{before} any other functions used within the same \texttt{paper} block. See [Automatic scaling to paper size], page 496.

Paper sizes are defined in \texttt{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 Section \texttt{A.5 [Predefined paper sizes]}, page 618.

The following command can be used in the file to add a custom paper size which can then be used with \texttt{set-default-paper-size} or \texttt{set-paper-size} as appropriate,
Chapter 4: Spacing issues

496

#(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.

See also

Notation Reference: [Automatic scaling to paper size], page 496, Section A.5 [Predefined paper sizes], page 618.

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], page 495.

The vertical dimensions affected by automatic scaling are top-margin and bottom-margin (see Section 4.1.3 [Fixed vertical spacing \paper variables], page 496). The horizontal dimensions affected by automatic scaling are left-margin, right-margin, inner-margin, outer-margin, binding-offset, indent, and short-indent (see Section 4.1.5 [Horizontal spacing \paper variables], page 499).

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.

See also

Notation Reference: Section 4.1.3 [Fixed vertical spacing \paper variables], page 496, Section 4.1.5 [Horizontal spacing \paper variables], page 499.

Installed Files: 'ly/paper-defaults-init.ly', 'scm/paper.scm'.

4.1.3 Fixed vertical spacing \paper variables

Note: Some \paper dimensions are automatically scaled to the paper size, which may lead to unexpected behavior. See [Automatic scaling to paper size], page 496.
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.

**See also**

Notation Reference: [Automatic scaling to paper size], page 496.


Snippets: Section “Spacing” in Snippets.

**Known issues and warnings**

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 Section 4.4.1 [Flexible vertical spacing within systems], page 515.

**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 skyslines) 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 Section 5.3.6 [Modifying alists], page 573. 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 Section 3.1.5 [File structure], page 446). 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.

See also

Notation Reference: Section 4.4.1 [Flexible vertical spacing within systems], page 515.
Snippets: Section “Spacing” in Snippets.

4.1.5 Horizontal spacing \paper variables

Note: Some \paper dimensions are automatically scaled to the paper size, which may lead to unexpected behavior. See [Automatic scaling to paper size], page 496.

\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

The horizontal extent of the staff lines in unindented, non-ragged systems, equal to (\paper-width − \left-margin − \right-margin) when unset. If \line-width is set, and both \left-margin and \right-margin are unset, then the margins will be updated to center the systems on the page automatically. Also see check-consistency. This variable can also be set in a \layout block.

\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 
(paper-width − line-width − right-margin). If only line-width is set, then 
both margins are set to ((paper-width − 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 dimen-
sion’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 
(paper-width − line-width − left-margin). If only line-width is set, then 
both margins are set to ((paper-width − line-width) / 2), and the systems 
are consequently centered on the page. Also see check-consistency.

check-consistency
If set to true, 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 its default value (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.

See also
Notation Reference: [Automatic scaling to paper size], page 496.

Installed Files: 'ly/paper-defaults-init.ly'.

Known issues and warnings
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.

See also
Notation Reference: [Automatic scaling to paper size], page 496.

\paper variables for shifts and indents
Default values (before scaling) that are not listed here are defined in

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. This variable can also be set in
a \layout block.

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. This
variable can also be set in a \layout block.

See also
Notation Reference: [Automatic scaling to paper size], page 496.
Snippets: Section “Spacing” in Snippets.

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

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.

See also
Notation Reference: Section 4.3.1 [Line breaking], page 507.
\paper variables for page breaking

Default values not listed here are defined in 'ly/paper-defaults-init.ly'

\section{page-breaking}


\section{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.

\section{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 Section 4.3.4 [Optimal page turning], page 510).

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.

\section{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.

\section{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.

\section{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.

\section{See also}

Notation Reference: Section 4.3.2 [Page breaking], page 509, Section 4.3.3 [Optimal page breaking], page 510, Section 4.3.4 [Optimal page turning], page 510, Section 4.3.5 [Minimal page breaking], page 511, Section 4.3.6 [One-line page breaking], page 511.

\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.

See also
Installed Files: 'ly/paper-defaults-init.ly'.

Known issues and warnings
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''' { c1 \break c1 \break c1 }
  }
}
Chapter 4: Spacing issues

See also

Installed Files: ‘ly/titling-init.ly’.
Snippets: Section “Spacing” in Snippets.

Known issues and warnings

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, Section 4.2.2 [Setting the staff size], page 506. Context modifications are discussed in a separate chapter; see Section 5.1.4 [Modifying context plug-ins], page 553 and Section 5.1.5 [Changing context default settings], page 555. The \paper variables that can appear in a \layout block are:

• line-width, ragged-right and ragged-last (see \paper variables for widths and margins, page 499)
• indent and short-indent (see \paper variables for shifts and indents, page 501)
• system-count (see \paper variables for line breaking, page 501)

Here is an example \layout block:

\layout {
indent = 2\cm
\context {
  \StaffGroup
    \override StaffGrouper.staff-staff-spacing.basic-distance = #8
  }
  \context {
    \Voice
}
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:
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

\begin{verbatim}
NoteHead.font-size = #4 \% (written in the variable definition)  
NoteHead.color = #red \% (added after the use of the variable)
\end{verbatim}

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.

**See also**

- Notation Reference: Section 5.1.5 [Changing context default settings], page 555.
- Snippets: Section “Spacing” in Snippets.

### 4.2.2 Setting the staff size

The default staff size is set to 20 points. This may be changed in two ways:

- 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.

\begin{verbatim}
#(set-global-staff-size 14)
\end{verbatim}

This sets the global default size to 14pt staff height and scales all fonts accordingly.

- To set the staff size individually for each score, use

\begin{verbatim}
\score{  
...  
\layout {  
  #(layout-set-staff-size 15)  
}  
}
\end{verbatim}

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:

<table>
<thead>
<tr>
<th>font name</th>
<th>staff height (pt)</th>
<th>staff height (mm)</th>
<th>use</th>
</tr>
</thead>
<tbody>
<tr>
<td>feta11</td>
<td>11.22</td>
<td>3.9</td>
<td>pocket scores</td>
</tr>
<tr>
<td>feta13</td>
<td>12.60</td>
<td>4.4</td>
<td></td>
</tr>
<tr>
<td>feta14</td>
<td>14.14</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>feta16</td>
<td>15.87</td>
<td>5.6</td>
<td></td>
</tr>
<tr>
<td>feta18</td>
<td>17.82</td>
<td>6.3</td>
<td></td>
</tr>
<tr>
<td>feta20</td>
<td>20</td>
<td>7.0</td>
<td></td>
</tr>
<tr>
<td>feta23</td>
<td>22.45</td>
<td>7.9</td>
<td></td>
</tr>
<tr>
<td>feta26</td>
<td>25.2</td>
<td>8.9</td>
<td></td>
</tr>
</tbody>
</table>

These fonts are available in any sizes. The context property \texttt{fontSize} and the layout property \texttt{staff-space} (in Section “StaffSymbol” in Internals Reference) can be used to tune the size for individual staves. The sizes of individual staves are relative to the global size.
See also
Notation Reference: [Selecting notation font size], page 204.
Snippets: Section “Spacing” in Snippets.

Known issues and warnings
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:
\begin{verbatim}
c4 c c c | \break
c4 c c c |
\end{verbatim}
\begin{music}
\begin{staff}
\relative c' {
\up \tuplet 3/2 { c4 c c } c2. |
\break
s1 |
\end{music}

By default, a \break in the middle of a measure is ignored, and a warning is printed. To force a line break in the middle of a measure, add an invisible bar line with \bar "":
\begin{verbatim}
c4 c c \bar "" \break
c |
c4 c c c |
\end{verbatim}
\begin{music}
\begin{staff}
\relative c' {
\up \tuplet 3/2 { c4 c c } c2 |
\bar "" \break
s1 |
\end{music}

A \break occurring at a bar line is also ignored if the previous measure ends in the middle of a note, such as when a tuplet begins and ends in different measures. To allow \break commands to work in these situations, remove the Forbid_line_break_engraver from the Voice context. Note that manually forced line breaks have to be added in parallel with the music:
\begin{verbatim}
\new Voice \with {
\remove "Forbid_line_break_engraver"
} \relative c' {
<<
{ c2. \tuplet 3/2 { c4 c c } c2. | }
{ s1 | \break s1 | }
\end{verbatim}
Similarly, line breaks are normally forbidden when beams cross bar lines. This behavior can be changed by setting \override Beam.breakable = ##t:

\override Beam.breakable = ##t
c2. c8[ c | \break
c8 c] c2. |

The \noBreak command forbids a line break at the bar line where it is inserted.

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... }
>>
Predefined commands
\break, \noBreak.

See also
Notation Reference: \[\text{\texttt{paper variables for line breaking}}\], page 501.
Snippets: Section “Spacing” in \textit{Snippets}.
Internals Reference: Section “LineBreakEvent” in \textit{Internals Reference}.

4.3.2 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. 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.

There are also analogous settings to \texttt{ragged-right} and \texttt{ragged-last} which have the same effect on vertical spacing. If \texttt{ragged-bottom} is set to \texttt{#t} the systems will not be justified vertically. When \texttt{ragged-last-bottom} is set to \texttt{#t}, as it is by default, empty space is allowed at the bottom of the final page (or the final page in each \texttt{bookpart}). See Section 4.1.3 [Fixed vertical spacing \texttt{\texttt{paper variables}}], page 496.

Page breaks are computed by the \texttt{page-breaking} function. LilyPond provides three algorithms for computing page breaks, \texttt{ly:optimal-breaking}, \texttt{ly:page-turnbreaking} and \texttt{ly:minimal-breaking}. The default is \texttt{ly:optimal-breaking}, but the value can be changed in the \texttt{\texttt{paper}} block:
\begin{verbatim}
\paper {
  page-breaking = #ly:page-turn-breaking
}
\end{verbatim}

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, \texttt{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.

\begin{verbatim}
\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"
  }
}
\end{verbatim}
Predefined commands
\pageBreak, \noPageBreak.

See also
Notation Reference: [\paper variables for page breaking], page 502.
Snippets: Section “Spacing” in Snippets.

4.3.3 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.

See also
Snippets: Section “Spacing” in Snippets.

4.3.4 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 Section 4.3.2 [Page breaking], page 509. 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, you can 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
The \texttt{Page\_turn\_engraver} detects volta repeats. It 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. The \texttt{Page\_turn\_engraver} can also disable page turns if the repeat is very short. If you set the context property \texttt{minimumRepeatLengthForPageTurn} then the \texttt{Page\_turn\_engraver} will only allow turns in repeats whose duration is longer than this value.

The page turning commands, \texttt{\pageTurn}, \texttt{\noPageTurn} and \texttt{\allowPageTurn}, may also be used at top-level, between scores and top-level markups.

\textbf{Predefined commands}

\texttt{\pageTurn}, \texttt{\noPageTurn}, \texttt{\allowPageTurn}.

\textbf{See also}

- Notation Reference: \texttt{[\textbackslash paper variables for line breaking]}, page 501.
- Snippets: Section “Spacing” in Snippets.

\textbf{Known issues and warnings}

There should only be one \texttt{Page\_turn\_engraver} in a score. If there is more than one, they will interfere with each other.

\textbf{4.3.5 Minimal page breaking}

The \texttt{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:

\begin{verbatim}
\paper {
  page-breaking = #ly:minimal-breaking
}
\end{verbatim}

\textbf{See also}

- Snippets: Section “Spacing” in Snippets.

\textbf{4.3.6 One-line page breaking}

The \texttt{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. This page breaking function does not typeset titles or margins; only the score will be displayed.

The page width will be adjusted so that the longest score fits on one line. In particular, \texttt{paper-width}, \texttt{line-width} and \texttt{indent} variables in the \texttt{\paper} block will be ignored, although \texttt{left-margin} and \texttt{right-margin} will still be honored. The height of the page will be left unmodified.

\textbf{4.3.7 Explicit breaks}

Lily sometimes rejects explicit \texttt{\break} and \texttt{\pageBreak} commands. There are two commands to override this behavior:

\begin{verbatim}
\override NonMusicalPaperColumn.line-break-permission = ##f
\override NonMusicalPaperColumn.page-break-permission = ##f
\end{verbatim}

When \texttt{line-break-permission} is overridden to false, Lily will insert line breaks at explicit \texttt{\break} commands and nowhere else. When \texttt{page-break-permission} is overridden to false, Lily will insert page breaks at explicit \texttt{\pageBreak} commands and nowhere else.
\paper {
    indent = #0
    ragged-right = ##t
    ragged-bottom = ##t
}

\score {
    \new Staff {
        \repeat unfold 2 { \music } \break
        \repeat unfold 4 { \music } \break
        \repeat unfold 6 { \music } \break
        \repeat unfold 8 { \music } \pageBreak
        \repeat unfold 8 { \music } \break
        \repeat unfold 6 { \music } \break
        \repeat unfold 4 { \music } \break
        \repeat unfold 2 { \music }
    }
    \layout {
        \context {
            \Score
            \override NonMusicalPaperColumn.line-break-permission = ##f
            \override NonMusicalPaperColumn.page-break-permission = ##f
        }
    }
}

\relative c'' {
    c8 c c c
}

\timesig.C44/clefs.G

\noteheads.s2

\noteheads.s2

\timesig.C44/clefs.G

\noteheads.s2

\timesig.C44/clefs.G

\noteheads.s2
4.3.8 Using an extra voice for breaks

Line- and page-breaking information usually appears within note entry directly.

\begin{verbatim}
music = \relative c'' { c4 c c c }
\score {
  \new Staff {
    \repeat unfold 2 { \music } \break
    \repeat unfold 3 { \music }
  }
}

This makes \break and \pageBreak commands easy to enter but mixes music entry with
information that specifies how music should lay out on the page. You can keep music entry
and line- and page-breaking information in two separate places by introducing an extra voice to
contain the breaks. This extra voice contains only skips together with \break, pageBreak and
other breaking layout information.

\begin{verbatim}
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 }
    }
  }
}
\end{verbatim}

See also

Snippets: Section “Spacing” in Snippets.
This pattern becomes especially helpful when overriding \texttt{line-break-system-details} and the other useful but long properties of \texttt{NonMusicalPaperColumnGrob}, as explained in Section 4.4 [Vertical spacing], page 515.

\begin{verbatim}
music = \relative c' { c4 c c c }

\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 * 2 \break
        \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details #'((Y-offset . 5)) s1 * 3 \break
        \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details #'((Y-offset . 15)) s1 * 6 \break
        \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details #'((Y-offset . 30)) s1 * 5 \break
      }
      \new Voice {
        \repeat unfold 2 { \music }
        \repeat unfold 3 { \music }
        \repeat unfold 6 { \music }
        \repeat unfold 5 { \music }
      }
    >>
  }
}
\end{verbatim}
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 Section 4.1.4 [Flexible vertical spacing \paper variables], page 497.

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:

<table>
<thead>
<tr>
<th>Non-staff line</th>
<th>Reference point</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChordNames</td>
<td>baseline</td>
</tr>
<tr>
<td>NoteNames</td>
<td>baseline</td>
</tr>
<tr>
<td>Lyrics</td>
<td>baseline</td>
</tr>
<tr>
<td>Dynamics</td>
<td>mid-height of 'm'</td>
</tr>
<tr>
<td>FiguredBass</td>
<td>highest point</td>
</tr>
<tr>
<td>FretBoards</td>
<td>top line</td>
</tr>
</tbody>
</table>

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 Section 4.1.4 [Flexible vertical spacing \paper variables], page 497. Specific methods for modifying alists are discussed in Section 5.3.6 [Modifying alists], page 573. 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:
Properties of the \texttt{VerticalAxisGroup} grob

\texttt{VerticalAxisGroup} properties are typically adjusted with an \texttt{\override} at the \texttt{Staff} level (or equivalent).

\texttt{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 \texttt{Lyrics}) are placed between the two staves. Does not apply to the bottom staff of a system.

Initially, the \texttt{staff-staff-spacing} of a \texttt{VerticalAxisGroup} is a Scheme function that applies the properties of the \texttt{StaffGrouper} if the staff is part of a group, or the \texttt{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 \texttt{override} shown above.

\texttt{default-staff-staff-spacing}

A flexible-spacing alist defining the \texttt{staff-staff-spacing} used for ungrouped staves, unless \texttt{staff-staff-spacing} has been explicitly set with an \texttt{\override}.

\texttt{staff-affinity}

The direction of the staff to use for spacing the current non-staff line. Choices are \texttt{UP}, \texttt{DOWN}, and \texttt{CENTER}. If \texttt{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 \texttt{staff-affinity} from top to bottom, e.g. a non-staff line set to \texttt{UP} should not immediately follow one that is set to \texttt{DOWN}. Non-staff lines at the top of a system should use \texttt{DOWN}; those at the bottom should use \texttt{UP}. Setting \texttt{staff-affinity} for a staff causes it to be treated as a non-staff line. Setting \texttt{staff-affinity} to \texttt{#f} causes a non-staff line to be treated as a staff. Setting \texttt{staff-affinity} to \texttt{UP}, \texttt{CENTER}, or \texttt{DOWN} causes a staff to be spaced as a non-staff line.

\texttt{nonstaff-relatedstaff-spacing}

The distance between the current non-staff line and the nearest staff in the direction of \texttt{staff-affinity}, if there are no non-staff lines between the two, and \texttt{staff-affinity} is either \texttt{UP} or \texttt{DOWN}. If \texttt{staff-affinity} is \texttt{CENTER}, then \texttt{nonstaff-relatedstaff-spacing} is used for the nearest staves on \textit{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 \texttt{stretchability} of one of these types of spacing to a small value will make that spacing dominate. Setting the \texttt{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.

See also
Notation Reference: Section 4.1.4 [Flexible vertical spacing \paper variables], page 497, Section 5.3.6 [Modifying alists], page 573.


Internals Reference: Section “Contexts” in Internals Reference, Section “VerticalAxisGroup” in Internals Reference, Section “StaffGrouper” in Internals Reference.

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], page 515.

  Additional properties are involved for staves that are part of a staff-group; see [Spacing of grouped staves], page 519.

  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.
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], page 515.

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 }

>>

\new StaffGroup <<
\new Staff { c'1 }
\new Staff { c'1 }

>>

>>
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 create the VerticalAxisGroup layout object.

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], page 515.

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 }
>>
```
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))

See also

Snippets: Section “Spacing” in Snippets.
Internals Reference: Section “Contexts” in Internals Reference, Section “VerticalAxisGroup” in Internals Reference.
\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 line- 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. See Section 4.3.8 [Using an extra voice for breaks], page 513.
Explicit \breaks evenly divide the music into six measures per line. Vertical spacing results from LilyPond’s defaults. To set the vertical startpoint of each system explicitly, we can set the \textit{Y-offset} pair in the \textit{line-break-system-details} attribute of the \texttt{NonMusicalPaperColumn} grob:

\begin{verbatim}
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
    \score {
        \mempty
        \slice { \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' }
        } >>
    }
}
\end{verbatim}
Note that \texttt{line-break-system-details} takes an associative list of potentially many values, but that we set only one value here. Note, too, that the \texttt{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 \texttt{alignment-distances} subproperty of \texttt{line-break-system-details}.

\begin{verbatim}
\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' }
  }
\end{verbatim}
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' } }
>>
>>
}
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.

See also

Snippets: Section “Spacing” in Snippets.

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.

\once \override TextScript.outside-staff-priority = #1
\once \override DynamicLineSpanner.outside-staff-priority = ##f
\once \override TextScript.outside-staff-padding = #0
\once \override TextScript.outside-staff-padding = #3
\once \override DynamicLineSpanner.outside-staff-padding = #3
\once \override TextScript.outside-staff-padding = #0

The vertical padding around outside-staff objects can be controlled with outside-staff-padding.

\once \override TextScript.outside-staff-padding = #0
\once \override TextScript.outside-staff-padding = #3
\once \override DynamicLineSpanner.outside-staff-padding = #3
\once \override TextScript.outside-staff-padding = #0
\once \override DynamicLineSpanner.outside-staff-padding = #0

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.

```
c4"Word" c c'2
R1
\once \override TextScript.outside-staff-horizontal-padding = #1
c,,4"Word" c c'2
```

See also

Snippets: Section “Spacing” in Snippets.

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 Section “SpacingSpanner” in Internals Reference object). The longer the duration, the more space it gets: doubling a duration adds a fixed amount (this amount is controlled by `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.

```
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. The shortest duration is printed when you run lilypond with the `--verbose` option.

These durations may also be customized. If you set the `common-shortest-duration` in Section “SpacingSpanner” in Internals Reference, 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:

```
c2 c4 | c4. c16 | c4. c8 | c8 c c c c
```

In the Essay on automated music engraving, it was explained that stem directions influence spacing (see Section “Optical spacing” in Essay). This is controlled with the `stem-spacing-correction` property in the Section “NoteSpacing” in Internals Reference, object. These are generated for every Section “Voice” in Internals Reference context. The `StaffSpacing` object (generated in Section “Staff” in Internals Reference context) contains the same property for controlling the stem/bar line spacing. The following example shows these corrections, once with default settings, and once with exaggerated corrections:

```
c2 c4 | c4. c16 | c4. c8 | c8 c c c c
```

Proportional notation is supported; see Section 4.5.5 [Proportional notation], page 535.

See also


Snippets: Section “Spacing” in Snippets.

Internals Reference: Section “SpacingSpanner” in Internals Reference, Section “NoteSpacing” in Internals Reference, Section “StaffSpacing” in Internals Reference, Section “NonMusicalPaperColumn” in Internals Reference.

Known issues and warnings

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 area

New sections with different spacing parameters can be started with `newSpacingSection`. This is useful when there are sections with a different notions of long and short notes.

In the following example, the time signature change introduces a new section, and hence the 16ths notes are automatically spaced slightly wider.

```latex
\time 2/4
c4 c8 c
c8 c4 c16[ c c8] c4
\newSpacingSection
\time 4/16
c16[ c c8]
```

The `\newSpacingSection` command creates a new `SpacingSpanner` object at that musical moment. 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. They will then affect the spacing of all the following music until the properties are changed in a new spacing section, for example,

```latex
\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]
```

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`.

```latex
\score {
  \relative c'' {
    g4 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 |
  }
}
```

See also

Snippets: Section “Spacing” in Snippets.

Internals Reference: Section “SpacingSpanner” in Internals Reference.
\score { 
  \relative c'' { 
    g4 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) 
  } 
}
Selected Snippets

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 {
  \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 { c8[ c \clef alto c \grace { c16 c } c8 c c c 32[ c ]}
```

See also

Snippets: Section “Spacing” in Snippets.

4.5.4 Line length

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 \texttt{ragged-last} is similar to \texttt{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.

\begin{verbatim}
\layout {
    indent = #0
    line-width = #150
    ragged-last = ##t
}
\end{verbatim}

**See also**

See Snippets: Section “Spacing” in \textit{Snippets}.

### 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:

- \texttt{proportionalNotationDuration}
- \texttt{uniform-stretching}
- \texttt{strict-note-spacing}
- \texttt{\remove "Separating_line_group_engraver"}
- \texttt{\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.

\begin{verbatim}
\score {
    \new RhythmicStaff {
      c'2 c'16 c' c' c' \tuplet 5/4 { c'16 c' c' c' c' }
    }
}
\end{verbatim}

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 {
    c'2 c'16 c' c' \uplet 5/4 { c'16 c' c' c' c' }
  }
  \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 {
    c'2 c'16 c' c' \uplet 5/4 { c'16 c' c' c' c' }
  }
  \layout {
    \context {
      \Score
      \proportionalNotationDuration = #(ly:make-moment 1/8)
    }
  }
}

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.
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 tuplets.

We start by examining what happens to our original example, with classical spacing, when we add a second staff with a different type of tuplet.

\score {
<<
  \new RhythmicStaff {
    c'2 c'16 c' c' \tuplet 5/4 { c'16 c' c' c' }
  }
>>
\layout {
  \context {\Score
    proportionalNotationDuration = #(ly:make-moment 1/16)
  }
}
}\score {
<<
  \new RhythmicStaff {
    c'2 c'16 c' c' \tuplet 5/4 { c'16 c' c' c' }
  }
>>
\layout {
  \context {\Score
    proportionalNotationDuration = #(ly:make-moment 1/32)
  }
}
}
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 { \tuplet 5/4 { c'16 c' c' c' } } \new RhythmicStaff { \tuplet 9/8 { c'8 c' c' c' c' c' c' c' } } >> }\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 { c'2 c'16 c' c' c' \tuplet 5/4 { c'16 c' c' c' } } \new RhythmicStaff { \tuplet 9/8 { c'8 c' c' c' c' c' c' c' } } >> }\layout { \context { \Score \uniform-stretching } } }
```
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 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 Section 4.5.2 [New spacing area], page 532, 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.
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''8 c'' c'' \clef alto d' d'2
}

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 c'' c'' \clef alto d' d'2
}

\new Staff {
  \set Score.proportionalNotationDuration = #(ly:make-moment 1/16)
  \override Score.SpacingSpanner.strict-note-spacing = ##t
  c''8 c'' c'' \clef alto d' d'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.

**See also**

Notation Reference: Section 4.5.2 [New spacing area], page 532.

Snippets: Section “Spacing” in Snippets.

### 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, Section 4.6.1 [Displaying spacing], page 541.

#### 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:

```latex
\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{align*}
1 \text{ point} & = (25.4/72.27) \text{ mm} \\
1 \text{ staff-space} & = \left(\frac{\text{staff-size}}{4}\right) \text{ pts} \\
& = \left(\frac{\text{staff-size}}{4}\right) \times \left(\frac{25.4}{72.27}\right) \text{ mm}
\end{align*}

In this case, one staff-space is approximately equal to 1.757 mm. 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.

See also

Notation Reference: Section 4.2.2 [Setting the staff size], page 506.
Snippets: Section “Spacing” in Snippets.

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 Section 4.1 [Page layout], page 494.

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.

\begin{verbatim}
\paper {
  system-system-spacing = #'((basic-distance . 0.1) (padding . 0))
  ragged-last-bottom = ##f
  ragged-bottom = ##f
}
\end{verbatim}
• 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 repeats (or alternate repeats) require extra space. If these repeats are spread over two systems, they will take up more space than one system with the volta repeats and another system without. For example, dynamics that ‘stick out’ of a system can be moved closer to the staff:

e4 c g\f c
e4 c g\tweak X-offset #-2.7 \f c

• Alter the horizontal spacing via SpacingSpanner. For more details, see Section 4.5.3 [Changing horizontal spacing], page 532. The following example illustrates the default spacing:

\score {
  \relative c'' {
    g4 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 c'' {
    g4 e e2 |
  }
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.

**See also**

Notation Reference: Section 4.1 [Page layout], page 494, Section 4.5.3 [Changing horizontal spacing], page 532.

Snippets: Section “Spacing” in *Snippets.*
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 Section “Tweaking output” in Learning Manual. 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: Section “the Internals Reference” in Internals Reference. That manual lists all the variables, functions and options available in LilyPond. It is written as a HTML document, which is available on-line, 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.

See also


Snippets: Section “Contexts and engravers” in Snippets.

Internals Reference: Section “Contexts” in Internals Reference, Section “Engravers and Performers” in Internals Reference.

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], page 555).

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], page 167) or creating music quotes ([Quoting other voices], page 195).

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 Section “Scheme tutorial” in Extending, contains a short tutorial on entering numbers, lists, strings, and symbols in Scheme.
[Automatic syllable durations], page 243. 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.

See also


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], page 198. 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 {
  \% leave the Voice context to be created implicitly
  c4 c
}
\new Staff {
  d4 d
```
and to place several voices into one staff:

```
\new Staff <<
  \new Voice { 
    \voiceOne 
    c8 c c4 c c 
  } 
  \new Voice { 
    \voiceTwo 
    g4 g g g 
  } 
>>
```

\texttt{\new} should always be used to specify unnamed contexts.

The difference between \texttt{\new} and \texttt{\context} is in the action taken:

- \texttt{\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" { 
    \voiceOne 
    c8 c c4 c c 
  } 
  \new Voice = "A" { 
    \voiceTwo 
    g4 g g g 
  } 
>>
```

- \texttt{\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:

\begin{verbatim}
\score {
  <<
    % score layout
    \new Staff <<
      \new Voice = "one" {
        \voiceOne
      }
      \new Voice = "two" {
        \voiceTwo
      }
    >>

    % musical content
    \context Voice = "one" {
      %relative c'' {
      c4 c c c
    }
    \context Voice = "two" {
      %relative c'' {
      g8 g g4 g g
    }
    
    >>
  }
\end{verbatim}
Alternatively, variables may be employed to similar effect. See Section “Organizing pieces with variables” in Learning Manual.

- `\context` with no name will match the first of any previously created contexts of the same type in the same context hierarchy, 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' {
  c1
  \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], page 243.

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], page 555. 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], page 558.

See also


Notation Reference: [Changing just one specific context], page 558, [Automatic syllable durations], page 243.

## 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: just one of the \Voice contexts in a Staff context or in a <<...>> construct will always persist to the end of the enclosing Staff context or <<...>> construct, even though there may be periods when it has nothing to do. The context to persist in this way will be the first one encountered in the first enclosed {...} construct, ignoring any in enclosed <<...>> constructs.

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 c'' { d4 d d d } \\
\musicB = \relative c'' { g4 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
  >>
}

\timesig.C44
\clef.G
\noteheads

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.
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 c' { a4 a a a } 
\accompaniment = \relative c' { d4 d d d } 
\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:
words = \lyricmode { These words fol -- low the mel -- o -- dy }

\score {
  \new Staff = "music" {
    \new Voice = "melody" {
      \voiceOne
      \melody
    }
    \new Voice = "accompaniment" {
      \voiceTwo
      \accompaniment
    }
  }
\lyricsto "melody" { \words }
}

These words follow the melo--dy

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 ↦ Translation ↦ Engravers. Every context described in Internals Reference ↦ Translation ↦ 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 {
    f2 g
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 Section “Visibility and color of objects” in Learning Manual.

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.

```latex
\score {
  <<
    \new Staff \with {
      \consists "Timing_translator"
      \consists "Default_bar_line_engraver"
    } \relative c'' {
      \time 3/4
      c4 c c c c c
    }
    \new Staff \with {
      \consists "Timing_translator"
      \consists "Default_bar_line_engraver"
    } \relative c'' {
      \time 2/4
      c4 c c c c c
    }
  >>
  \layout {
    \context {\Score \remove "Timing_translator" \remove "Default_bar_line_engraver"}
  }
}
```
Known issues and warnings

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.

See also

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 Section 5.3 [Modifying properties], page 567. 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], page 545).

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 c' {

• Directly setting a context property
\score {
  \relative c'' {
    a4"Smaller font" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Staff
      fontSize = #-4
    }
  }
}

Smaller font
\score {
  \relative c'' {
    a4"Dynamics above" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Voice
      \dynamicUp
    }
    \context {
      \Staff
      \accidentalStyle dodecaphonic
    }
  }
}

• A predefined command such as \dynamicUp or a music expression like \accidentalStyle dodecaphonic
\score {
  \relative c'' {
    a4"Dynamics above" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Voice
      \dynamicUp
    }
    \context {
      \Staff
      \accidentalStyle dodecaphonic
    }
  }
}
Dynamics above $\textit{ff}$

- A user-defined variable containing a `\with` block; for details of the `\with` block see [Changing just one specific context], page 558.

```
StaffDefaults = \with {
  fontSize = #-4
}
```

```
\score {
  \new Staff {
    \relative c'' {
      a4"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], page 547. 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 c'' {
      a4"Smaller font" a a a
      a4 a a a
    }
  }
  \layout {
    \accidentalStyle dodecaphonic
    \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 c'' {
    a4^"Thick stems" a a a
    a4 a a a
  }
}
}
```

```
Thick stems
\timesig.C44/clefs.G
```

- Directly setting a context property

```
\score {
<<
\new Staff {
  \relative c'' {
    a4^"Default font" a a a
    a4 a a a
  }
}
\new Staff \with { fontSize = #-4 }
{
  \relative c'' {
    a4^"Smaller font" a a a
    a4 a a a
  }
}
>>
}
```
A predefined command such as `\dynamicUp`

```latex
\score {
\new Staff {
\new Voice {
\relative c'' {
    a4^"Dynamics below" a a a
    a4 a a\ff a
}
}
}
\new Staff \with { \accidentalStyle dodecaphonic } {
\new Voice \with { \dynamicUp } {
\relative c'' {
    a4^"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.

See also

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,

```
\begin{music}
\noteheads.s0/\noteheads.s2\slash
\timesig.C44/\clefs.G/\noteheads.s2/\noteheads.s2\slash
\accidentals.flat/\noteheads.s2
\end{music}
```

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"
\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
}

The opposite of \accepts is \denies, which is sometimes needed when reusing existing context definitions.

Putting both into a \layout block, like
\layout {
\context {
\name ImproVoice
...
}
\context {
\Staff
\accepts "ImproVoice"
}
}

Then the output at the start of this subsection can be entered as
\relative c'' {
\new ImproVoice {
\c4"ad lib" c
\c4 c"undress"
\c c_"while playing :)"
}
\a1
}\relative c''

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.

See also
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 Section “Nesting music expressions” in Learning Manual, Section 1.6.2 [Modifying single staves], page 182 and Section 2.1.2 [Techniques specific to lyrics], page 251.

See also


Notation Reference: Section 1.6.2 [Modifying single staves], page 182, Section 2.1.2 [Techniques specific to lyrics], page 251.

Application Usage: Section “An extra staff appears” in Application Usage.

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
\stemUp
f

If you visit the documentation on fingering instructions (in [Fingering instructions], page 205), you will notice:

See also
Internals Reference: Section “Fingering” in Internals Reference.

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 Section “Fingering” in Internals Reference. At the top of the page, you will see

Fingering objects are created by: Section “Fingering_engraver” in Internals Reference and Section “New_fingering_engraver” in Internals Reference.

By following related links inside the program reference, we can follow the flow of information within the program:

- Section “Fingering” in Internals Reference: Section “Fingering” in Internals Reference objects are created by: Section “Fingering_engraver” in Internals Reference
- Section “Fingering_engraver” in Internals Reference: Music types accepted: Section “fingering-event” in Internals Reference
- Section “fingering-event” in Internals Reference: Music event type fingering-event is in Music expressions named Section “FingeringEvent” in Internals Reference

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 Section “Translation” in Internals Reference, and the Section “Backend” in Internals Reference. 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 Section “Fingering” in Internals Reference. 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 Section “grob-interface” in Internals Reference.

The page for Fingering lists the definitions for the Fingering object. For example, the page says padding (dimension, in staff space):
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 Section “Fingering” in Internals Reference page at the bottom

This object supports the following interfaces: Section “item-interface” in Internals Reference, Section “self-alignment-interface” in Internals Reference, Section “side-position-interface” in Internals Reference, Section “text-interface” in Internals Reference, Section “text-script-interface” in Internals Reference, Section “font-interface” in Internals Reference, Section “finger-interface” in Internals Reference, and Section “grob-interface” in Internals Reference.

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 Section “Other sources of information” in Learning Manual) ‘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
Since the 2 is vertically positioned next to its note, we have to meddle with the interface associated with this positioning. This is done using `side-position-interface`. The page for this interface says:

**side-position-interface**

Position a victim object (this one) next to other objects (the support). The property `direction` signifies where to put the victim object relative to the support (left or right, up or down?)

Below this description, the variable `padding` is described as:

- `padding` (dimension, in staff space)
  - Add this much extra space between objects that are next to each other.

By increasing the value of `padding`, we can move the fingering away from the note head. The following command inserts 3 staff spaces of white between the note and the fingering:

```
\once \override Voice.Fingering.padding = #3
```

Inserting this command before the Fingering object is created, i.e., before c2, yields the following result:

```
\once \override Voice.Fingering.padding = #3
c-2
\stemUp
f
```

In this case, the context for this tweak is `Voice`. This fact can also be deduced from the program reference, for the page for the Section “Fingering_engraver” in *Internals Reference* plug-in says:

Fingering_engraver is part of contexts: . . . Section “Voice” in *Internals Reference*

### 5.2.4 Naming conventions

Another thing that is needed, is an overview of the various naming conventions:

- scheme functions: lowercase-with-hyphens (incl. one-word names)
- scheme functions: ly:plus-scheme-style
- music events, music classes and music properties: as-scheme-functions
- Grob interfaces: scheme-style
- backend properties: scheme-style (but X and Y!)
- contexts (and MusicExpressions and grobs): Capitalized or CamelCase
- context properties: lowercaseFollowedByCamelCase
- engravers: Capitalized_followed_by_lower_case_and_with_underscores
Questions to be answered:

- Which of these are conventions and which are rules?
- Which are rules of the underlying language, and which are LP-specific?

5.3 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.

See also

Internals Reference: Section “Backend” in Internals Reference, Section “All layout objects” in Internals Reference, Section “OverrideProperty” in Internals Reference, Section “RevertProperty” in Internals Reference, Section “PropertySet” in Internals Reference.

Known issues and warnings

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. localKeySignature (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).
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.

\set \Score.autoBeaming = ##f
<<
{ e8 e e e
  \set autoBeaming = ##t
  e8 e e e
} \\ { c8 c c c c8 c c c}
>>

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
<<
{ \unset autoBeaming
e8 e e e
  \unset \Score.autoBeaming
e8 e e e
} \\ { c8 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 command by \once makes the setting apply to only a single time-step:
c4 \once \set fontSize = #4.7
c4

A full description of all available context properties is in the internals reference, see Translation ↦ Tunable context properties.

See also

Internals Reference: Section “Tunable context properties” in Internals Reference.

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:
c4 c \override Voice.Stem.thickness = #3.0
c4 c

If no context is specified in an \override, the bottom context is used:
{ \override Staff.Stem.thickness = #3.0 <<
    { e4 e \override Stem.thickness = #0.5 e4 e } \\ { c4 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,
c4
\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:
{
  <<
  {
    e4
    \override Staff.Stem.thickness = #3.0
    e4 e e
  } \ \ {
    c4 c c
    \revert Staff.Stem.thickness
    c4
  }
  >>
}

\once can be used with \override to affect only the current time step:
{
  <<
  {
    \override Stem.thickness = #3.0
    e4 e e e
  } \ \ {
    c4
  }

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 Section “Tweaking methods” in Learning Manual.

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:
\begin{verbatim}
< c
 \tweak color #red
d
g
 \tweak duration-log #1
a
> 4
\end{verbatim}

\tweak can be used to modify slurs:
\begin{verbatim}
c-\tweak thickness #5 ( d e f)
\end{verbatim}
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 c4
\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 c8 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 Section “Displaying music expressions” in `Extending`. 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.

See also


Extending LilyPond: Section “Displaying music expressions” in `Extending`. 
Known issues and warnings
The \texttt{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 \texttt{set} vs. \texttt{override}
Both \texttt{set} and \texttt{override} manipulate properties associated with contexts. In either case, properties heed the hierarchy of contexts: properties not set in a context itself show the values of the respective parent context.

Values and lifetime of context properties are dynamic and only available when music is being interpreted, ‘iterated’. At the time of context creation, properties are initialized from the corresponding context definition and possible context modifications. Afterwards, changes are achieved with property-setting commands in the music itself.

Now grob definitions are a special category of context properties. Since their structure, bookkeeping and use is different from ordinary context properties, they are accessed with a different set of commands, and treated separately in the documentation.

As opposed to plain context properties, grob definitions are subdivided into grob properties. A “grob” (graphical object) is usually created by an engraver at the time of interpreting a music expression and receives its initial properties from the current grob definition of the engraver’s context. The engraver (or other ‘backend’ parts of LilyPond) may subsequently add or change properties to the grob, but that does not affect the context’s grob definition.

What we call ‘grob properties’ in the context of user-level tweaking are actually the properties of a context’s grob definition. In contrast to ordinary context properties, grob definitions have the bookkeeping required to keep track of its parts, the individual grob properties (and even subproperties of them) separately so that it is possible to define those parts in different contexts and have the overall grob definition at the time of grob creation be assembled from pieces provided in different contexts among the current context and its parents.

Grob definitions are manipulated using \texttt{override} and \texttt{revert} and have a name starting with a capital letter (like ‘NoteHead’) whereas ordinary context properties are manipulated using \texttt{set} and \texttt{unset} and are named starting with a lowercase letter.

The special commands \texttt{tweak} and \texttt{override\texttt{Property} change grob properties bypassing context properties completely. Instead they catch grobs as they are being created and then directly set properties on them when they originate from a tweaked music event or are of a particular kind, respectively.

5.3.6 Modifying alists
Some user-configurable properties are internally represented as \textit{alists} (association lists), which store pairs of keys and values. The structure of an alist is:

\begin{verbatim}
'(((key1 . value1) (key2 . value2) (key3 . value3) ...) ...
\end{verbatim}

If an alist is a grob property or \texttt{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 \texttt{staff-staff-spacing} property of the \texttt{StaffGrouper} grob. The property is an alist with four keys: \texttt{basic-distance}, \texttt{minimum-distance}, \texttt{padding}, and \texttt{stretchability}. The standard settings for this property are listed in the “Backend” section of the Internals Reference (see Section “StaffGrouper” in \textit{Internals Reference}):
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:

\begin{verbatim}
\override \StaffGrouper.staff-staff-spacing = #'((basic-distance . 7))
\end{verbatim}

\begin{verbatim}
\override \StaffGrouper.staff-staff-spacing = #'((basic-distance . 7)
  (minimum-distance . 0)
  (padding . 0)
  (stretchability . 7))
\end{verbatim}

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.

\begin{note}
Nested declarations will not work for context property alists (such as `beamExceptions`, `keySignature`, `timeSignatureSettings`, etc.). These properties can only be modified by completely re-defining them as alists.
\end{note}

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.

\textbf{Chord mode}

This is activated with the `\chordmode` command, and causes input to be interpreted with the syntax of chord notation, see Section 2.7 [Chord notation], page 384. 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], page 389.

\textbf{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], page 362.

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], page 362.

\textbf{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], page 398.
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], page 398.

**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], page 318.

To create fret diagrams above a staff, you have two choices. You can either use the \FretBoards context (see [Automatic fret diagrams], page 353 or you can enter them as a markup above the notes using the `\fret-diagram` command (see [Fret diagram markups], page 334).

**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 Section 2.1 [Vocal music], page 239.

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 Section A.10 [Text markup commands], page 645.

**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.

For example, to indicate dynamic markings for the verses of a choral pieces it is necessary to enter note mode to interpret the markings:

```
{ c4 c4 c4 c4 }
\addlyrics {
  \notemode{\set stanza = \markup{ \dynamic f 1. } }
  To be sung loudly
}
\addlyrics {
  \notemode{\set stanza = \markup{ \dynamic p 2. } }
  To be sung quietly
}
```

```
\timesig{C44}
\clef{G}
\notehead{s2}
\f1. To be sung loudly
\p2. To be sung quietly
```
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:

\slurDown
\slurDown
\slurDown
\slurDown
\timesig C44 /clefs G /timesig C44 /clefs G /timesig C44 /clefs G /timesig C44 /

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 Section “Within-staff objects” in Learning Manual.

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.
In polyphonic music, it is generally better to specify an explicit voice than change an object’s direction. For more information. See Section 1.5.2 [Multiple voices], page 159.

See also


Notation Reference: Section 1.5.2 [Multiple voices], page 159.

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 Section 4.1 [Page layout], page 494.

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 Section 4.2.2 [Setting the staff size], page 506.

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 Section “Length and thickness of objects” in Learning Manual.

See also


Notation Reference: Section 4.1 [Page layout], page 494, Section 4.2.2 [Setting the staff size], page 506.

5.4.4 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.

Note: 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.
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
}
{ a4 e' f b | d1 }

5.4.5 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

\-\tweak minimum-length #5

\a

\a1
\compressFullBarRests
R1*23
% increase the length of the rest bar
\once \override MultiMeasureRest.minimum-length = #20
R1*23
a1

% 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:
\tweak minimum-length #5
(a g) (g)
\tweak minimum-length #5
(a (g) \)

For some layout objects, the \textit{minimum-length} property becomes effective only if the \textit{set-spacing-rods} procedure is called explicitly. To do this, the \textit{springs-and-rods} property should be set to \texttt{ly:spanner::set-spacing-rods}. For example, the minimum length of a glissando has no effect unless the \textit{springs-and-rods} property is set:
% default
\glissando c'
% not effective alone
\once \override Glissando.minimum-length = #20
\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
\glissando c'
The same is true of the \texttt{Beam} object:

\begin{verbatim}
\% not effective alone
\override Beam.minimum-length = #20
\once e8 e e e
\% effective only when both overrides are present
\override Beam.minimum-length = #20
\override Beam.springs-and-rods = #ly:spanner::set-spacing-rods
e8 e e e
\end{verbatim}

![Example Beam notation]

\textit{The to-barline property}

The second useful property of the \texttt{spanner-interface} is \texttt{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:

\begin{verbatim}
a \textless a a a a \textgreater ! a a a \break
\override Hairpin.to-barline = ##f
a \textless a a a a \textgreater ! a a a
\end{verbatim}

![Example Hairpin notation]

This property is not effective for all spanners. For example, setting it to \texttt{#t} has no effect on slurs or phrasing slurs or on other spanners for which terminating on the bar line would not be meaningful.

\textbf{Using the line-spanner-interface}

Objects which support the \texttt{line-spanner-interface} include

- \texttt{DynamicTextSpanner}
- \texttt{Glissando}
- \texttt{TextSpanner}
- \texttt{TrillSpanner}
- \texttt{VoiceFollower}

The routine responsible for drawing the stencils for these spanners is \texttt{ly:line-interface::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 \texttt{\override} command is quite simple:
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
c1 \glissando \break
f1

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.
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:

\begin{verbatim}
\override TextSpanner.bound-details.left.stencil-align-dir-y = #-2
\override TextSpanner.bound-details.right.stencil-align-dir-y = #UP
\end{verbatim}

Note that negative values move the text up, contrary to the effect that might be expected, as a value of -1 or DOWN means align the bottom edge of the text with the spanner line. A value of 1 or UP aligns the top edge of the text with the spanner line.

arrow

Setting this sub-property to #t produces an arrowhead at the end of the line.

padding

This sub-property controls the space between the specified end point of the line and the actual end. Without padding, a glissando would start and end in the center of each note head.

The music function $\texttt{\endSpanners}$ 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.

\begin{verbatim}
\endSpanners
c2 \startTextSpan c2 c2
\endSpanners
c2 \< c2 c2
\end{verbatim}

When using $\texttt{\endSpanners}$ it is not necessary to close $\texttt{\startTextSpan}$ with $\texttt{\stopTextSpan}$, nor is it necessary to close hairpins with $\texttt{\!}$. 
See also


5.4.6 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 Section “Visibility and color of objects” in Learning Manual.

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 \textit{#f} no function will be called and the object will not be drawn. The default action can be recovered with \texttt{\revert}.

\begin{verbatim}
\override Score.BarLine.stencil = ##f
\revert Score.BarLine.stencil
\end{verbatim}

This rather common operation has a shortcut \texttt{\omit}:

\begin{verbatim}
\omit Score.BarLine
\undo \omit Score.BarLine
\end{verbatim}

Making objects transparent

Every layout object has a transparent property which by default is set to \texttt{#f}. If set to \texttt{#t} the object still occupies space but is made invisible.

\begin{verbatim}
\once \override NoteHead.transparent = ##t
\end{verbatim}

This rather common operation has a shortcut \texttt{\hide}:
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
```

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
```

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 Section “Visibility and color of objects” in Learning Manual. 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:

<table>
<thead>
<tr>
<th>Function form</th>
<th>Vector form</th>
<th>Before break</th>
<th>At no break</th>
<th>After break</th>
</tr>
</thead>
<tbody>
<tr>
<td>all-visible</td>
<td>'##(#t #t #t)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>begin-of-line-visible</td>
<td>'##(#f #f #t)</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>center-visible</td>
<td>'##(#f #t #f)</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>end-of-line-visible</td>
<td>'##(#t #f #f)</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>begin-of-line-invisible</td>
<td>'##(#t #f #f)</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>center-invisible</td>
<td>'##(#t #f #t)</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>end-of-line-invisible</td>
<td>'##(#f #t #f)</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>all-invisible</td>
<td>'##(#f #f #f)</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Layout object</th>
<th>Usual context</th>
<th>Default setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>BarLine</td>
<td>Score</td>
<td>calculated</td>
</tr>
<tr>
<td>BarNumber</td>
<td>Score</td>
<td>begin-of-line-visible</td>
</tr>
<tr>
<td>BreathingSign</td>
<td>Voice</td>
<td>begin-of-line-invisible</td>
</tr>
<tr>
<td>Clef</td>
<td>Staff</td>
<td>begin-of-line-visible</td>
</tr>
<tr>
<td>Custos</td>
<td>Staff</td>
<td>end-of-line-visible</td>
</tr>
<tr>
<td>DoublePercentRepeat</td>
<td>Voice</td>
<td>begin-of-line-invisible</td>
</tr>
<tr>
<td>KeyCancellation</td>
<td>Staff</td>
<td>begin-of-line-invisible</td>
</tr>
<tr>
<td>KeySignature</td>
<td>Staff</td>
<td>begin-of-line-visible</td>
</tr>
<tr>
<td>ClefModifier</td>
<td>Staff</td>
<td>begin-of-line-visible</td>
</tr>
<tr>
<td>RehearsalMark</td>
<td>Score</td>
<td>end-of-line-invisible</td>
</tr>
<tr>
<td>TimeSignature</td>
<td>Staff</td>
<td>all-visible</td>
</tr>
</tbody>
</table>

The example below shows the use of the vector form to control the visibility of bar lines:

```
\once \override Score.BarLine.break-visibility = ##(#f #t #t)
\break
f4 g a b
```

\[ \begin{align*}
    \text{\music{f4 g a b}} & \quad \text{\music{f4 g a b}} \\
\end{align*} \]
Although all three components of the vector used to override `break-visibility` must be present, not all of them are effective with every layout object, and some combinations may even give errors. The following limitations apply:

- Bar lines cannot be printed at start of line.
- A bar number cannot be printed at the start of the first line unless it is set to be different from 1.
- Clef – see below
- Double percent repeats are either all printed or all suppressed. Use `begin-of-line-invisible` to print and `all-invisible` to suppress.
- Key signature – see below
- ClefModifier – see below

**Special considerations**

**Visibility following explicit changes**

The `break-visibility` property controls the visibility of key signatures and changes of clef only at the start of lines, i.e. after a break. It has no effect on the visibility of the key signature or clef following an explicit key change or an explicit clef change within or at the end of a line. In the following example the key signature following the explicit change to B-flat major is still visible, even though `all-invisible` is set.

\key g \major
f4 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.
Visibility of cancelling accidentals

To remove the cancelling accidentals printed at an explicit key change, set the Staff context property `printKeyCancellation` to `#f`:

\key g \major
f4 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

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:
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:

```
\key g \major
f4 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.

See also
5.4.7 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 \texttt{ly:line-interface::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.

\begin{verbatim}
d2 \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
\end{verbatim}

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:

\begin{verbatim}
e2 \glissando f
\once \override Glissando.bound-details.right.Y = #-2
e2 \glissando f
\end{verbatim}

The value for \texttt{Y} is set to \texttt{-2} for the right end point. The left side may be similarly adjusted by specifying \texttt{left} instead of \texttt{right}.

If \texttt{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 Section 5.4.5 [Spanners], page 579.

5.4.8 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 \texttt{grob-interface} can be rotated by setting their \texttt{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:

\override Hairpin.rotation = #'(20 -1 0)

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

5.5 Advanced tweaks

This section discusses various approaches to fine tuning the appearance of the printed score.

See also


Notation Reference: Section 5.2 [Explaining the Internals Reference], page 564, Section 5.3 [Modifying properties], page 567.

Extending LilyPond: Section “Interfaces for programmers” in Extending.

Installed Files: ‘scm/define-grobs.scm’.

Snippets: Section “Tweaks and overrides” in Snippets.

Internals Reference: Section “All layout objects” in Internals Reference.

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 Section “self-alignment-interface” in Internals Reference and Section “side-position-interface” in Internals Reference.
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.

**Note:** 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.

**See also**

Notation Reference: [Using the break-alignable-interface], page 594.

Extending LilyPond: Section “Callback functions” in Extending.

**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
```

![Example noteheads]

**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:

<table>
<thead>
<tr>
<th>side-axis property</th>
<th>direction property</th>
<th>Placement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>left</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>right</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>below</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>above</td>
</tr>
</tbody>
</table>

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"
```

right-aligned
center-aligned
left-aligned
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

\timesig.C44/clefs.G 3/noteheads.s2

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 ""
\e1

\% 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 ""
\e2.

\% 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
\e1
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.

```latex
% 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 ""
e1
% 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.

```latex
% 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 ""
e1
% The RehearsalMark will be centered above the Key Signature
\once \override Score.KeySignature.break-align-anchor-alignment = #CENTER
```
\mark ""
\key a \major
e1
% 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 ""
e1

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
\once \override Score.RehearsalMark.break-align-symbols = #'(key-signature)
\once \override Score.KeySignature.break-align-anchor = #3.5
\key a \major
\mark ""
e1
% 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 ""
e1

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], page 230.

The following example demonstrates this by changing the note head symbol to a cross within a circle.

\一次性 \override NoteHead.stencil = #ly:text-interface::print
\一次性 \override NoteHead.text = \markup {
    \combine
    \halign #-0.7 \draw-circle #0.85 #0.2 ##f
    \musicglyph #'noteheads.s2cross
}

\relative c'' {
    a a \XinO a a
}

Any of the glyphs in the feta Font can be supplied to the \musicglyph markup command – see Section A.8 [The Feta font], page 624.

See also

Notation Reference: [Graphic notation inside markup], page 230, Section 1.8.2 [Formatting text], page 223, Section A.10 [Text markup commands], page 645, Section A.8 [The Feta font], page 624.

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:

a. by specifying the displacements to be made to the control points of the automatically calculated Bézier curve, or

b. 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.

```
{ e1~ e }
\`
{ 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
  e1~ e
}
```

```
```

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:
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:

\[ 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:

\[ c4( f g c \break d,4 c' f, c) \]

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
\[ \shape #'((0 . 0) (0 . 0) (0 . 0) (0 . 1)) \]
\[ \shape #'((0 . 0) (0 . 0) (0 . 0) (0 . 0)) \]
\[ c4( f g c \break d,4 c' f, c) \]
If an S-shaped curve is required the control points must always be adjusted manually — LilyPond will never select such shapes automatically.

\(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.

c1
\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,
}
}
Known issues and warnings

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.

See also

Internals Reference: Section “TieColumn” in Internals Reference.

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.

\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:

r2
\alterBroken thickness #'(10 1) Slur
c8( d e f
\break
The \texttt{\textbackslash alterBroken} command may be used with any spanner object, including \texttt{Tie}, \texttt{PhrasingSlur}, \texttt{Beam} and \texttt{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 \texttt{\textbackslash tweak} form of the command:

\begin{verbatim}
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
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.

**Note:** 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
```
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.

**Note:** 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 Section “Music functions” in Extending.

#### 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
   (parser location arg1 arg2 ...) 
   (type1? type2? ...) 
   #
   ...
   #})
```

where

- `argN` is the `n`th argument
- `typeN?` is a scheme *type predicate* for which `argN` must return `#t`.
- `...music...` is normal LilyPond input, using `$` (in places where only Lilypond constructs are allowed) or `#` (to use it as a Scheme value or music function argument or music inside of music lists) to reference arguments (e.g. `'#arg1').

The `parser` and `location` arguments are mandatory, and are used in some advanced situations as described in the ‘Extending’ manual (see Section “Music functions” in Extending). For substitution functions, just be sure to include them.

The list of type predicates is also 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 Section A.20 [Predefined type predicates], page 744. User-defined type predicates are also allowed.

See also
Notation Reference: Section A.20 [Predefined type predicates], page 744.
Extending Lilypond: Section “Music functions” in Extending.

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
    (parser location padding)
    (number?)
    
      \once \override TextScript.padding = #padding
    #)

\relative c'' {
  c4^"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
    (parser location note)
    (ly:music?)
```
Substitution functions with multiple arguments can be defined:

```music
\( \text{tempoPadded} = \) {
    (define-music-function
        (parser location padding tempotext)
        (number? markup?)
        {
            \once \override Score.MetronomeMark.padding = #padding
            \tempo \markup { \bold #tempotext }
        }
    )
}
```

```music
\relative c'' { 
    \tempo \markup { "Low tempo" }
    c4 d e f g1
    \tempoPadded #4.0 "High tempo"
    g4 f e d c1
}
```

High tempo
Appendix 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.

<table>
<thead>
<tr>
<th>Ignatzek (default)</th>
<th>Alternative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignatzek (default)</td>
<td>Alternative</td>
</tr>
<tr>
<td>C</td>
<td>Cm</td>
</tr>
<tr>
<td>C</td>
<td>C♭3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Def</th>
<th>Alt</th>
</tr>
</thead>
<tbody>
<tr>
<td>C⁷</td>
<td>Cm⁷</td>
</tr>
<tr>
<td>C⁷</td>
<td>C⁷ b³</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Def</th>
<th>Alt</th>
</tr>
</thead>
<tbody>
<tr>
<td>C⁷ #⁵</td>
<td>Cm∆</td>
</tr>
<tr>
<td>C⁷ #⁵</td>
<td>C♭³ #⁷</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Def</th>
<th>Alt</th>
</tr>
</thead>
<tbody>
<tr>
<td>C⁶</td>
<td>Cm⁶</td>
</tr>
<tr>
<td>C⁶</td>
<td>C♭³ 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Def</th>
<th>Alt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cm¹³</td>
<td>Cm¹¹</td>
</tr>
<tr>
<td>Cm¹³ b³</td>
<td>C¹¹ b³</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Def</th>
<th>Alt</th>
</tr>
</thead>
<tbody>
<tr>
<td>C⁷ #⁹</td>
<td>C¹¹</td>
</tr>
<tr>
<td>C⁷ #⁹</td>
<td>C¹¹</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Def</th>
<th>Alt</th>
</tr>
</thead>
<tbody>
<tr>
<td>C⁷ #¹¹ b¹³</td>
<td>C⁷ #⁵ #⁹</td>
</tr>
<tr>
<td>C⁷ #¹¹ b¹³</td>
<td>C⁷ #⁵ #⁹</td>
</tr>
</tbody>
</table>
### A.2 Common chord modifiers

The following table shows chord modifiers that can be used to generate standard chord structures.

<table>
<thead>
<tr>
<th>Type</th>
<th>Interval</th>
<th>Modifier</th>
<th>Example</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major</td>
<td>Major third, perfect fifth</td>
<td>5 or nothing</td>
<td>c1:5</td>
<td><img src="#" alt="C" /></td>
</tr>
<tr>
<td>Minor</td>
<td>Minor third, perfect fifth</td>
<td>m or m5</td>
<td>c1:m</td>
<td><img src="#" alt="Cm" /></td>
</tr>
<tr>
<td>Notation</td>
<td>Description</td>
<td>Symbol</td>
<td>Abbreviation</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>Augmented</td>
<td>Major third, augmented fifth</td>
<td>aug</td>
<td>c1:aug</td>
<td></td>
</tr>
<tr>
<td>Diminished</td>
<td>Minor third, diminished fifth</td>
<td>dim</td>
<td>c1:dim</td>
<td></td>
</tr>
<tr>
<td>Dominant seventh</td>
<td>Major triad, minor seventh</td>
<td>7</td>
<td>c1:7</td>
<td></td>
</tr>
<tr>
<td>Major seventh</td>
<td>Major triad, major seventh</td>
<td>maj7 or maj</td>
<td>c1:maj7</td>
<td></td>
</tr>
<tr>
<td>Minor seventh</td>
<td>Minor triad, minor seventh</td>
<td>m7</td>
<td>c1:m7</td>
<td></td>
</tr>
<tr>
<td>Diminished seventh</td>
<td>Diminished triad, diminished seventh</td>
<td>dim7</td>
<td>c1:dim7</td>
<td></td>
</tr>
<tr>
<td>Augmented seventh</td>
<td>Augmented triad, minor seventh</td>
<td>aug7</td>
<td>c1:aug</td>
<td></td>
</tr>
<tr>
<td>Half-diminished seventh</td>
<td>Diminished triad, minor seventh</td>
<td>m7.5-</td>
<td>c1:m7.5-</td>
<td></td>
</tr>
<tr>
<td>Minor-major seventh</td>
<td>Minor triad, major seventh</td>
<td>m7+</td>
<td>m7+</td>
<td></td>
</tr>
<tr>
<td>Notation</td>
<td>Description</td>
<td>Number</td>
<td>Symbol</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------</td>
<td>--------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>Major sixth</td>
<td>Major triad, sixth</td>
<td>6</td>
<td>C6</td>
<td></td>
</tr>
<tr>
<td>Minor sixth</td>
<td>Minor triad, sixth</td>
<td>m6</td>
<td>Cm6</td>
<td></td>
</tr>
<tr>
<td>Dominant ninth</td>
<td>Dominant seventh, major ninth</td>
<td>9</td>
<td>C9</td>
<td></td>
</tr>
<tr>
<td>Major ninth</td>
<td>Major seventh, major ninth</td>
<td>maj9</td>
<td>Cmaj9</td>
<td></td>
</tr>
<tr>
<td>Minor ninth</td>
<td>Minor seventh, major ninth</td>
<td>m9</td>
<td>Cm9</td>
<td></td>
</tr>
<tr>
<td>Dominant eleventh</td>
<td>Dominant ninth, perfect eleventh</td>
<td>11</td>
<td>C11</td>
<td></td>
</tr>
<tr>
<td>Major eleventh</td>
<td>Major ninth, perfect eleventh</td>
<td>maj11</td>
<td>Cmaj11</td>
<td></td>
</tr>
<tr>
<td>Minor eleventh</td>
<td>Minor ninth, perfect eleventh</td>
<td>m11</td>
<td>Cm11</td>
<td></td>
</tr>
<tr>
<td>Notation</td>
<td>Description</td>
<td>Number</td>
<td>Prefix</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------</td>
<td>--------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>Dominant thirteenth</td>
<td>Dominant ninth, major thirteenth</td>
<td>13</td>
<td>c1:13</td>
<td></td>
</tr>
<tr>
<td>Dominant eleventh, major thirteenth</td>
<td>13.11</td>
<td>c1:13.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Major thirteenth</td>
<td>Major eleventh, major thirteenth</td>
<td>maj13.11</td>
<td>c1:maj13.11</td>
<td></td>
</tr>
<tr>
<td>Minor thirteenth</td>
<td>Minor eleventh, major thirteenth</td>
<td>m13.11</td>
<td>c1:m13.11</td>
<td></td>
</tr>
<tr>
<td>Suspended second</td>
<td>Major second, perfect fifth</td>
<td>sus2</td>
<td>c1:sus2</td>
<td></td>
</tr>
<tr>
<td>Suspended fourth</td>
<td>Perfect fourth, perfect fifth</td>
<td>sus4</td>
<td>c1:sus4</td>
<td></td>
</tr>
<tr>
<td>Power chord (two-voiced)</td>
<td>Perfect fifth</td>
<td>1.5</td>
<td>\powerChords c1:1.5</td>
<td></td>
</tr>
<tr>
<td>Power chord (three-voiced)</td>
<td>Perfect fifth, octave</td>
<td>1.5.8</td>
<td>\powerChords c1:1.5.8</td>
<td></td>
</tr>
</tbody>
</table>

### A.3 Predefined string tunings

The chart below shows the predefined string tunings.
Guitar tunings

<table>
<thead>
<tr>
<th>Guitar tuning</th>
<th>Guitar seven-string-tuning</th>
<th>Guitar drop-d-tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Guitar tuning" /></td>
<td><img src="image" alt="Guitar seven-string-tuning" /></td>
<td><img src="image" alt="Guitar drop-d-tuning" /></td>
</tr>
</tbody>
</table>

Bass tunings

<table>
<thead>
<tr>
<th>Bass tuning</th>
<th>Bass four-string-tuning</th>
<th>Bass drop-d-tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Bass tuning" /></td>
<td><img src="image" alt="Bass four-string-tuning" /></td>
<td><img src="image" alt="Bass drop-d-tuning" /></td>
</tr>
</tbody>
</table>

Mandolin tunings

<table>
<thead>
<tr>
<th>Mandolin tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Mandolin tuning" /></td>
</tr>
</tbody>
</table>

Banjo tunings

<table>
<thead>
<tr>
<th>Banjo tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Banjo tuning" /></td>
</tr>
</tbody>
</table>

Ukulele tunings

<table>
<thead>
<tr>
<th>Ukulele tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Ukulele tuning" /></td>
</tr>
</tbody>
</table>
A.4 Predefined fretboard diagrams

Diagrams for Guitar
Diagrams for Ukulele

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>Cm</th>
<th>C+</th>
<th>C°</th>
<th>C△</th>
<th>Cm7</th>
<th>C7</th>
<th>Csus2</th>
<th>Csus4</th>
<th>C9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>123</td>
<td>1 4</td>
<td>1324</td>
<td>1</td>
<td>1111</td>
<td>122</td>
<td>13</td>
<td>2 1</td>
<td></td>
</tr>
<tr>
<td>C#</td>
<td>C#m</td>
<td>C#+</td>
<td>C#°</td>
<td>C#△</td>
<td>C#m7</td>
<td>C#7</td>
<td>C#sus2</td>
<td>C#sus4</td>
<td>C#9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1114</td>
<td>1233</td>
<td>2114</td>
<td>1 2</td>
<td>1112</td>
<td>1113</td>
<td>2213</td>
<td>1111</td>
<td>1233</td>
<td>1312</td>
</tr>
<tr>
<td>Db</td>
<td>Dbm</td>
<td>Db+</td>
<td>Dbo</td>
<td>Db7</td>
<td>Db△</td>
<td>Dbm7</td>
<td>Dbsus2</td>
<td>Dbsus4</td>
<td>Db9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1114</td>
<td>1233</td>
<td>2114</td>
<td>1 2</td>
<td>1112</td>
<td>1113</td>
<td>2213</td>
<td>1111</td>
<td>1233</td>
<td>1312</td>
</tr>
<tr>
<td>D</td>
<td>Dm</td>
<td>D+</td>
<td>D°</td>
<td>D△</td>
<td>Dm7</td>
<td>D7</td>
<td>Dsus2</td>
<td>Dsus4</td>
<td>D9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>123</td>
<td>221</td>
<td>2114</td>
<td>1324</td>
<td>1112</td>
<td>1113</td>
<td>2213</td>
<td>1111</td>
<td>12</td>
<td>1312</td>
</tr>
<tr>
<td>D#</td>
<td>D#m</td>
<td>D#+</td>
<td>D#°</td>
<td>D#△</td>
<td>D#m7</td>
<td>D#7</td>
<td>D#sus2</td>
<td>D#sus4</td>
<td>D#9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>221</td>
<td>3321</td>
<td>221</td>
<td>1314</td>
<td>1112</td>
<td>1212</td>
<td>2214</td>
<td>1111</td>
<td>2211</td>
<td>1341</td>
</tr>
<tr>
<td>Eb</td>
<td>Ebm</td>
<td>Eb+</td>
<td>Eb°</td>
<td>Eb△</td>
<td>Ebm7</td>
<td>Eb7</td>
<td>Esus2</td>
<td>Esus4</td>
<td>Eb9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>221</td>
<td>3321</td>
<td>221</td>
<td>1314</td>
<td>1112</td>
<td>1212</td>
<td>2214</td>
<td>1111</td>
<td>2211</td>
<td>1341</td>
</tr>
<tr>
<td>E</td>
<td>Em</td>
<td>E+</td>
<td>E°</td>
<td>E△</td>
<td>Em7</td>
<td>E7</td>
<td>Esus2</td>
<td>Esus4</td>
<td>E9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2341</td>
<td>3321</td>
<td>1 4</td>
<td>1 2</td>
<td>12 3</td>
<td>13 2</td>
<td>1 2</td>
<td>1111</td>
<td>24 1</td>
<td>1222</td>
</tr>
<tr>
<td>F</td>
<td>Fm</td>
<td>F+</td>
<td>F°</td>
<td>F△</td>
<td>Fm7</td>
<td>F7</td>
<td>Fsus2</td>
<td>Fsus4</td>
<td>F9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>24</td>
<td>2114</td>
<td>1324</td>
<td>2314</td>
<td>2413</td>
<td>1324</td>
<td>2214</td>
<td>13</td>
</tr>
<tr>
<td>F#</td>
<td>F#m</td>
<td>F#+</td>
<td>F#°</td>
<td>F#△</td>
<td>F#m7</td>
<td>F#7</td>
<td>F#sus2</td>
<td>F#sus4</td>
<td>F#9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3423</td>
<td>2341</td>
<td>3421</td>
<td>3424</td>
<td>2413</td>
<td>1324</td>
<td>2294</td>
<td>1424</td>
<td>1423</td>
<td>1222</td>
</tr>
</tbody>
</table>
Appendix A: Notation manual tables

<table>
<thead>
<tr>
<th>GB</th>
<th>GBm</th>
<th>GB+</th>
<th>GB°</th>
<th>GB7</th>
<th>GB△</th>
<th>GBm7</th>
<th>GB6</th>
<th>GBsus2</th>
<th>GBsus4</th>
<th>GB9</th>
</tr>
</thead>
<tbody>
<tr>
<td>[diagram]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GB</td>
<td>Gm</td>
<td>G+</td>
<td>G°</td>
<td>G7</td>
<td>G△</td>
<td>Gm7</td>
<td>G6</td>
<td>Gsus2</td>
<td>Gsus4</td>
<td>G9</td>
</tr>
<tr>
<td>[diagram]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GB#</td>
<td>GB#m</td>
<td>GB#+</td>
<td>GB#°</td>
<td>GB#7</td>
<td>GB#△</td>
<td>GB#m7</td>
<td>GB#6</td>
<td>GB#sus2</td>
<td>GB#sus4</td>
<td>GB#9</td>
</tr>
<tr>
<td>[diagram]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ab</td>
<td>Abm</td>
<td>Ab+</td>
<td>Ab°</td>
<td>Ab7</td>
<td>Ab△</td>
<td>Abm7</td>
<td>Ab6</td>
<td>Absus2</td>
<td>Absus4</td>
<td>Ab9</td>
</tr>
<tr>
<td>[diagram]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Am</td>
<td>A+</td>
<td>A°</td>
<td>A7</td>
<td>A△</td>
<td>Am7</td>
<td>A6</td>
<td>Asus2</td>
<td>Asus4</td>
<td>A9</td>
</tr>
<tr>
<td>[diagram]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A#</td>
<td>A#m</td>
<td>A#+</td>
<td>A#°</td>
<td>A#7</td>
<td>A#△</td>
<td>A#m7</td>
<td>A#6</td>
<td>A#sus2</td>
<td>A#sus4</td>
<td>A#9</td>
</tr>
<tr>
<td>[diagram]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bb</td>
<td>Bbm</td>
<td>Bb+</td>
<td>Bb°</td>
<td>Bb7</td>
<td>Bb△</td>
<td>Bbm7</td>
<td>Bb6</td>
<td>Bsus2</td>
<td>Bsus4</td>
<td>Bb9</td>
</tr>
<tr>
<td>[diagram]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Bm</td>
<td>B+</td>
<td>B°</td>
<td>B7</td>
<td>B△</td>
<td>Bm7</td>
<td>B6</td>
<td>Bsus2</td>
<td>Bsus4</td>
<td>B9</td>
</tr>
<tr>
<td>[diagram]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagrams for Mandolin

<table>
<thead>
<tr>
<th>C</th>
<th>Cm</th>
<th>C+</th>
<th>C°7</th>
<th>C7</th>
<th>C△</th>
<th>Cm7</th>
<th>CØ</th>
<th>C6</th>
<th>Csus2</th>
<th>Csus4</th>
<th>C9</th>
</tr>
</thead>
<tbody>
<tr>
<td>[diagram]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix A: Notation manual tables

<table>
<thead>
<tr>
<th></th>
<th>C#</th>
<th>C#m</th>
<th>C#+</th>
<th>C#7</th>
<th>C#△</th>
<th>C#m^7</th>
<th>C#∅</th>
<th>C#6</th>
<th>C#sus2</th>
<th>C#sus4</th>
<th>C#9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ⅳ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4231</td>
<td>231</td>
<td>41</td>
<td>41</td>
<td>211</td>
<td>4213</td>
<td>1122</td>
<td>1122</td>
<td>1134</td>
<td>3111</td>
<td>2134</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Db</th>
<th>Dbm</th>
<th>Db+</th>
<th>Db^7</th>
<th>Db△</th>
<th>Dbm^7</th>
<th>Db∅</th>
<th>Db6</th>
<th>Db^sus2</th>
<th>Db^sus4</th>
<th>Db^9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ⅳ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4231</td>
<td>231</td>
<td>41</td>
<td>41</td>
<td>211</td>
<td>4213</td>
<td>1122</td>
<td>1122</td>
<td>1134</td>
<td>3111</td>
<td>2134</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>Dm</th>
<th>D+</th>
<th>D^7</th>
<th>D△</th>
<th>Dm^7</th>
<th>D∅</th>
<th>D^6</th>
<th>D^sus2</th>
<th>D^sus4</th>
<th>D^9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>12</td>
<td>1</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>12</td>
<td>1</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>D#</th>
<th>D#m</th>
<th>D#+</th>
<th>D#^7</th>
<th>D#△</th>
<th>D#m^7</th>
<th>D#∅</th>
<th>D#6</th>
<th>D#^sus2</th>
<th>D#^sus4</th>
<th>D#^9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3114</td>
<td>3112</td>
<td>123</td>
<td>2143</td>
<td>2143</td>
<td>2143</td>
<td>3142</td>
<td>3142</td>
<td>1134</td>
<td>3111</td>
<td>2134</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Eb</th>
<th>Ebm</th>
<th>Eb+</th>
<th>Eb^7</th>
<th>Eb△</th>
<th>Ebm^7</th>
<th>Eb∅</th>
<th>Eb6</th>
<th>Eb^sus2</th>
<th>Eb^sus4</th>
<th>Eb^9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3114</td>
<td>3112</td>
<td>123</td>
<td>2143</td>
<td>2143</td>
<td>2143</td>
<td>3142</td>
<td>3142</td>
<td>1134</td>
<td>3111</td>
<td>2134</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>E</th>
<th>Em</th>
<th>E+</th>
<th>E^7</th>
<th>E△</th>
<th>Em^7</th>
<th>E∅</th>
<th>E^6</th>
<th>E^sus2</th>
<th>E^sus4</th>
<th>E^9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>23</td>
<td>234</td>
<td>2143</td>
<td>1</td>
<td>2</td>
<td>112</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>132</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td></td>
<td></td>
<td>3111</td>
<td>3111</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>Fm</th>
<th>F+</th>
<th>F^7</th>
<th>F△</th>
<th>Fm^7</th>
<th>F∅</th>
<th>F^6</th>
<th>F^sus2</th>
<th>F^sus4</th>
<th>F^9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>1341</td>
<td>1341</td>
<td>1</td>
<td>32</td>
<td>231</td>
<td>1111</td>
<td>2</td>
<td>341</td>
<td>4211</td>
<td></td>
</tr>
<tr>
<td>1341</td>
<td>1341</td>
<td>1341</td>
<td>1341</td>
<td>1</td>
<td>32</td>
<td>231</td>
<td>1111</td>
<td>2</td>
<td>341</td>
<td>4211</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>F#</th>
<th>F#m</th>
<th>F#+</th>
<th>F#^7</th>
<th>F#△</th>
<th>F#m^7</th>
<th>F#∅</th>
<th>F#6</th>
<th>F#^sus2</th>
<th>F#^sus4</th>
<th>F#^9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2341</td>
<td>1341</td>
<td>1341</td>
<td>1341</td>
<td>1</td>
<td>32</td>
<td>231</td>
<td>1111</td>
<td>2</td>
<td>341</td>
<td>4211</td>
<td></td>
</tr>
<tr>
<td>2341</td>
<td>1341</td>
<td>1341</td>
<td>1341</td>
<td>1</td>
<td>32</td>
<td>231</td>
<td>1111</td>
<td>2</td>
<td>341</td>
<td>4211</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Gb</th>
<th>Gbm</th>
<th>Gb+</th>
<th>Gb^7</th>
<th>Gb△</th>
<th>Gbm^7</th>
<th>Gb∅</th>
<th>Gb6</th>
<th>Gb^sus2</th>
<th>Gb^sus4</th>
<th>Gb^9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td>Ⅲ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2341</td>
<td>1341</td>
<td>1341</td>
<td>1341</td>
<td>1</td>
<td>32</td>
<td>231</td>
<td>1111</td>
<td>2</td>
<td>341</td>
<td>4211</td>
<td></td>
</tr>
<tr>
<td>2341</td>
<td>1341</td>
<td>1341</td>
<td>1341</td>
<td>1</td>
<td>32</td>
<td>231</td>
<td>1111</td>
<td>2</td>
<td>341</td>
<td>4211</td>
<td></td>
</tr>
</tbody>
</table>
### 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)
<table>
<thead>
<tr>
<th>Notation</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>a5</td>
<td>148 x 210 mm</td>
</tr>
<tr>
<td>a4</td>
<td>210 x 297 mm</td>
</tr>
<tr>
<td>a3</td>
<td>297 x 420 mm</td>
</tr>
<tr>
<td>a2</td>
<td>420 x 594 mm</td>
</tr>
<tr>
<td>a1</td>
<td>594 x 841 mm</td>
</tr>
<tr>
<td>a0</td>
<td>841 x 1189 mm</td>
</tr>
</tbody>
</table>

The “ISO 216” B Series

<table>
<thead>
<tr>
<th>Notation</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>b10</td>
<td>31 x 44 mm</td>
</tr>
<tr>
<td>b9</td>
<td>44 x 62 mm</td>
</tr>
<tr>
<td>b8</td>
<td>62 x 88 mm</td>
</tr>
<tr>
<td>b7</td>
<td>88 x 125 mm</td>
</tr>
<tr>
<td>b6</td>
<td>125 x 176 mm</td>
</tr>
<tr>
<td>b5</td>
<td>176 x 250 mm</td>
</tr>
<tr>
<td>b4</td>
<td>250 x 353 mm</td>
</tr>
<tr>
<td>b3</td>
<td>353 x 500 mm</td>
</tr>
<tr>
<td>b2</td>
<td>500 x 707 mm</td>
</tr>
<tr>
<td>b1</td>
<td>707 x 1000 mm</td>
</tr>
<tr>
<td>b0</td>
<td>1000 x 1414 mm</td>
</tr>
</tbody>
</table>

Two extended sizes as defined in “DIN 476”

<table>
<thead>
<tr>
<th>Notation</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>4a0</td>
<td>1682 x 2378 mm</td>
</tr>
<tr>
<td>2a0</td>
<td>1189 x 1682 mm</td>
</tr>
</tbody>
</table>

“ISO 269” standard C series

<table>
<thead>
<tr>
<th>Notation</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>c10</td>
<td>28 x 40 mm</td>
</tr>
<tr>
<td>c9</td>
<td>40 x 57 mm</td>
</tr>
<tr>
<td>c8</td>
<td>57 x 81 mm</td>
</tr>
<tr>
<td>c7</td>
<td>81 x 114 mm</td>
</tr>
<tr>
<td>c6</td>
<td>114 x 162 mm</td>
</tr>
<tr>
<td>c5</td>
<td>162 x 229 mm</td>
</tr>
<tr>
<td>c4</td>
<td>229 x 324 mm</td>
</tr>
<tr>
<td>c3</td>
<td>324 x 458 mm</td>
</tr>
<tr>
<td>c2</td>
<td>458 x 648 mm</td>
</tr>
<tr>
<td>c1</td>
<td>648 x 917 mm</td>
</tr>
<tr>
<td>c0</td>
<td>917 x 1297 mm</td>
</tr>
</tbody>
</table>

North American paper sizes

<table>
<thead>
<tr>
<th>Notation</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>junior-legal</td>
<td>8.0 x 5.0 in</td>
</tr>
<tr>
<td>legal</td>
<td>8.5 x 14.0 in</td>
</tr>
</tbody>
</table>
"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.

<table>
<thead>
<tr>
<th>Acoustic Grand</th>
<th>Contrabass</th>
<th>Lead 7 (fifths)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bright Acoustic</td>
<td>Tremolo Strings</td>
<td>Lead 8 (bass+lead)</td>
</tr>
<tr>
<td>Electric Grand</td>
<td>Pizzicato Strings</td>
<td>Pad 1 (new age)</td>
</tr>
<tr>
<td>Honky-tonk</td>
<td>Orchestral Harp</td>
<td>Pad 2 (warm)</td>
</tr>
<tr>
<td>Electric Piano 1</td>
<td>Timpani</td>
<td>Pad 3 (polysynth)</td>
</tr>
<tr>
<td>Electric Piano 2</td>
<td>String Ensemble 1</td>
<td>Pad 4 (choir)</td>
</tr>
<tr>
<td>Harpsichord</td>
<td>String Ensemble 2</td>
<td>Pad 5 (bowed)</td>
</tr>
<tr>
<td>Clav</td>
<td>Synthstrings 1</td>
<td>Pad 6 (metallic)</td>
</tr>
<tr>
<td>Celesta</td>
<td>Synthstrings 2</td>
<td>Pad 7 (halo)</td>
</tr>
<tr>
<td>Glockenspiel</td>
<td>Choir Aahs</td>
<td>Pad 8 (sweep)</td>
</tr>
<tr>
<td>Music Box</td>
<td>Voice Oohs</td>
<td>Fx 1 (rain)</td>
</tr>
<tr>
<td>Vibraphone</td>
<td>Synth Voice</td>
<td>Fx 2 (soundtrack)</td>
</tr>
<tr>
<td>Marimba</td>
<td>Orchestra Hit</td>
<td>Fx 3 (crystal)</td>
</tr>
<tr>
<td>Xylophone</td>
<td>Trumpet</td>
<td>Fx 4 (atmosphere)</td>
</tr>
<tr>
<td>Tubular Bells</td>
<td>Trombone</td>
<td>Fx 5 (brightness)</td>
</tr>
<tr>
<td>Dulcimer</td>
<td>Tuba</td>
<td>Fx 6 (goblins)</td>
</tr>
<tr>
<td>Drawbar Organ</td>
<td>Muted Trumpet</td>
<td>Fx 7 (echoes)</td>
</tr>
<tr>
<td>Percussive Organ</td>
<td>French Horn</td>
<td>Fx 8 (sci-fi)</td>
</tr>
<tr>
<td>Rock Organ</td>
<td>Brass Section</td>
<td>Sitar</td>
</tr>
<tr>
<td>Church Organ</td>
<td>Synthbrass 1</td>
<td>Banjo</td>
</tr>
<tr>
<td>Reed Organ</td>
<td>Synthbrass 2</td>
<td>Shamisen</td>
</tr>
<tr>
<td>Accordion</td>
<td>Soprano Sax</td>
<td>Koto</td>
</tr>
<tr>
<td>Harmonica</td>
<td>Alto Sax</td>
<td>Kalimba</td>
</tr>
<tr>
<td>Concertina</td>
<td>Tenor Sax</td>
<td>Bagpipe</td>
</tr>
<tr>
<td>Acoustic Guitar (nylon)</td>
<td>Baritone Sax</td>
<td>Fiddle</td>
</tr>
<tr>
<td>Acoustic Guitar (steel)</td>
<td>Oboe</td>
<td>Shanai</td>
</tr>
<tr>
<td>Electric Guitar (jazz)</td>
<td>English Horn</td>
<td>Tinkle Bell</td>
</tr>
<tr>
<td>Electric Guitar (clean)</td>
<td>Bassoon</td>
<td>Agogo</td>
</tr>
<tr>
<td>Electric Guitar (muted)</td>
<td>Clarinet</td>
<td>Steel Drums</td>
</tr>
<tr>
<td>Overdriven Guitar</td>
<td>Piccolo</td>
<td>Woodblock</td>
</tr>
<tr>
<td>Distorted Guitar</td>
<td>Flute</td>
<td>Taiko Drum</td>
</tr>
<tr>
<td>Guitar Harmonics</td>
<td>Recorder</td>
<td>Melodic Tom</td>
</tr>
<tr>
<td>Acoustic Bass</td>
<td>Pan Flute</td>
<td>Synth Drum</td>
</tr>
<tr>
<td>Electric Bass (finger)</td>
<td>Blown Bottle</td>
<td>Reverse Cymbal</td>
</tr>
<tr>
<td>Electric Bass (pick)</td>
<td>Shakuhachi</td>
<td>Guitar Fret Noise</td>
</tr>
<tr>
<td>Fretless Bass</td>
<td>Whistle</td>
<td>Breath Noise</td>
</tr>
<tr>
<td>Slap Bass 1</td>
<td>Ocarina</td>
<td>Seashore</td>
</tr>
<tr>
<td>Slap Bass 2</td>
<td>Lead 1 (square)</td>
<td>Bird Tweet</td>
</tr>
<tr>
<td>Synth Bass 1</td>
<td>Lead 2 (sawtooth)</td>
<td>Telephone Ring</td>
</tr>
<tr>
<td>Synth Bass 2</td>
<td>Lead 3 (calliope)</td>
<td>Helicopter</td>
</tr>
<tr>
<td>Violin</td>
<td>Lead 4 (chiff)</td>
<td>Applause</td>
</tr>
<tr>
<td>Viola</td>
<td>Lead 5 (charang)</td>
<td>Gunshot</td>
</tr>
<tr>
<td>Cello</td>
<td>Lead 6 (voice)</td>
<td></td>
</tr>
</tbody>
</table>

### A.7 List of colors
Normal colors

Usage syntax is detailed in [Coloring objects], page 208.

<table>
<thead>
<tr>
<th>black</th>
<th>white</th>
<th>red</th>
<th>green</th>
</tr>
</thead>
<tbody>
<tr>
<td>blue</td>
<td>cyan</td>
<td>magenta</td>
<td>yellow</td>
</tr>
<tr>
<td>grey</td>
<td>darkred</td>
<td>darkgreen</td>
<td>darkblue</td>
</tr>
<tr>
<td>darkcyan</td>
<td>darkmagenta</td>
<td>darkyellow</td>
<td></td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>snow</th>
<th>GhostWhite</th>
<th>WhiteSmoke</th>
<th>gainsboro</th>
<th>FloralWhite</th>
</tr>
</thead>
<tbody>
<tr>
<td>OldLace</td>
<td>linen</td>
<td>AntiqueWhite</td>
<td>Papaya Whip</td>
<td>BlanchedAlmond</td>
</tr>
<tr>
<td>bisque</td>
<td>PeachPuff</td>
<td>Navajo White</td>
<td>moccasin</td>
<td>cornsilk</td>
</tr>
<tr>
<td>ivory</td>
<td>LemonChiffon</td>
<td>seashell</td>
<td>honeydew</td>
<td>MintCream</td>
</tr>
<tr>
<td>azure</td>
<td>AliceBlue</td>
<td>lavender</td>
<td>Lavender Blush</td>
<td>Misty Rose</td>
</tr>
<tr>
<td>white</td>
<td>black</td>
<td>DarkSlate Grey</td>
<td>Dim Grey</td>
<td>Slate Grey</td>
</tr>
<tr>
<td>LightSlateGrey</td>
<td>grey</td>
<td>Light Grey</td>
<td>Midnight Blue</td>
<td>navy</td>
</tr>
<tr>
<td>NavyBlue</td>
<td>CornflowerBlue</td>
<td>Dark Slate Blue</td>
<td>Slate Blue</td>
<td>Medium Slate Blue</td>
</tr>
<tr>
<td>LightSlateBlue</td>
<td>Medium Blue</td>
<td>Royal Blue</td>
<td>blue</td>
<td>Dodger Blue</td>
</tr>
<tr>
<td>DeepSkyBlue</td>
<td>SkyBlue</td>
<td>Light Sky Blue</td>
<td>Steel Blue</td>
<td>Light Steel Blue</td>
</tr>
<tr>
<td>LightBlue</td>
<td>PowderBlue</td>
<td>Pale Turquoise</td>
<td>Dark Turquoise</td>
<td>Medium Turquoise</td>
</tr>
<tr>
<td>turquoise</td>
<td>cyan</td>
<td>Light Cyan</td>
<td>Cadet Blue</td>
<td>Medium Aquamarine</td>
</tr>
<tr>
<td>aquamarine</td>
<td>Dark Green</td>
<td>Dark Olive Green</td>
<td>Dark Sea Green</td>
<td>Sea Green</td>
</tr>
<tr>
<td>MediumSeaGreen</td>
<td>Light Sea Green</td>
<td>Pale Green</td>
<td>Spring Green</td>
<td>Lawn Green</td>
</tr>
<tr>
<td>green</td>
<td>chartreuse</td>
<td>Medium Spring Green</td>
<td>Green Yellow</td>
<td>Lime Green</td>
</tr>
<tr>
<td>YellowGreen</td>
<td>Forest Green</td>
<td>Olive Drab</td>
<td>Dark Khaki</td>
<td>khaki</td>
</tr>
<tr>
<td>PaleGoldenrod</td>
<td>Light GoldenrodYellow</td>
<td>Light Yellow</td>
<td>yellow</td>
<td>gold</td>
</tr>
<tr>
<td>LightGoldenrod</td>
<td>goldenrod</td>
<td>Dark Goldenrod</td>
<td>Rosy Brown</td>
<td>Indian Red</td>
</tr>
<tr>
<td>SaddleBrown</td>
<td>sienna</td>
<td>peru</td>
<td>burly wood</td>
<td>beige</td>
</tr>
<tr>
<td>wheat</td>
<td>Sandy Brown</td>
<td>tan</td>
<td>chocolate</td>
<td>firebrick</td>
</tr>
<tr>
<td>brown</td>
<td>Dark Salmon</td>
<td>salmon</td>
<td>Light Salmon</td>
<td>orange</td>
</tr>
<tr>
<td>DarkOrange</td>
<td>coral</td>
<td>Light Coral</td>
<td>tomato</td>
<td>Orange Red</td>
</tr>
<tr>
<td>red</td>
<td>Hot Pink</td>
<td>Deep Pink</td>
<td>pink</td>
<td>Light Pink</td>
</tr>
<tr>
<td>PaleVioletRed</td>
<td>maroon</td>
<td>Medium Violet Red</td>
<td>Violet Red</td>
<td>magenta</td>
</tr>
<tr>
<td>violet</td>
<td>plum</td>
<td>orchid</td>
<td>Medium Orchid</td>
<td>Dark Orchid</td>
</tr>
<tr>
<td>DarkViolet</td>
<td>Blue Violet</td>
<td>purple</td>
<td>Medium Purple</td>
<td>thistle</td>
</tr>
<tr>
<td>DarkGrey</td>
<td>Dark Blue</td>
<td>Dark Cyan</td>
<td>Dark Magenta</td>
<td>Dark Red</td>
</tr>
<tr>
<td>LightGreen</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Color names with a numerical suffix

In the following names the suffix N can be a number in the range 1-4:

<table>
<thead>
<tr>
<th>snowN</th>
<th>seashellN</th>
<th>AntiqueWhiteN</th>
<th>bisqueN</th>
<th>PeachPuffN</th>
</tr>
</thead>
<tbody>
<tr>
<td>NavajoWhiteN</td>
<td>LemonChiffonN</td>
<td>cornsilkN</td>
<td>ivoryN</td>
<td>honeydewN</td>
</tr>
<tr>
<td>LavenderBlushN</td>
<td>MistyRoseN</td>
<td>azureN</td>
<td>SlateBlueN</td>
<td>RoyalBlueN</td>
</tr>
<tr>
<td>blueN</td>
<td>DodgerBlueN</td>
<td>SteelBlueN</td>
<td>DeepSkyBlueN</td>
<td>SkyBlueN</td>
</tr>
</tbody>
</table>
Appendix A: Notation Manual Tables

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 Section 1.8.2 [Formatting text], page 223.

Clefs glyphs

\clefs.C \clefs.C_change
\clefs.F \clefs.F_change
\clefs.G \clefs.G_change
\clefs.percussion \clefs.percussion_change
\clefs.tab \clefs.tab_change

Time Signature glyphs

timesig.C44  \timesig.C22
Number glyphs

plus  +  comma ,

hyphen -  period .

zero  0  one  1

two  2  three  3

four  4  five  5

six  6  seven  7

eight  8  nine  9

Accidental glyphs

accidentals.sharp  #  accidentals .sharp.arrowup  

accidentals .sharp.arrowdown  #  accidentals .sharp.arrowboth  

accidentals.sharp .slashslash.stem  #  accidentals.sharp .slashslashlashslash.stemstem  

accidentals.sharp .slashslashlashslash.stemstem  #  accidentals.sharp .slashslash.stemstemstem  

accidentals.natural  ♭  accidentals .natural.arrowup  

accidentals .natural.arrowdown  ♭  accidentals .natural.arrowboth  

accidentals.flat  ♮  accidentals.flat.arrowup  

accidentals.flat.arrowboth  

accidentals.flat.arrowdown
Appendix A: Notation manual tables

### Accidentals

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>accidentals.flat.arrowdown</td>
<td>Flat accidental down</td>
</tr>
<tr>
<td>accidentals.flat.arrowboth</td>
<td>Flat accidental both</td>
</tr>
<tr>
<td>accidentals.flat.slash</td>
<td>Flat slash</td>
</tr>
<tr>
<td>accidentals.mirroredflat.flat</td>
<td>Mirrored flat flat</td>
</tr>
<tr>
<td>accidentals.mirroredflat.backslash</td>
<td>Mirrored flat backslash</td>
</tr>
<tr>
<td>accidentals.flatflat.slash</td>
<td>Flat flat slash</td>
</tr>
</tbody>
</table>

### Default Notehead Glyphs

<table>
<thead>
<tr>
<th>Notehead</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>noteheads.uM2</td>
<td>Upright medium 2</td>
</tr>
<tr>
<td>noteheads.dM2</td>
<td>Downright medium 2</td>
</tr>
<tr>
<td>noteheads.sM1</td>
<td>Square medium 1</td>
</tr>
<tr>
<td>noteheads.s0</td>
<td>Square 0</td>
</tr>
<tr>
<td>noteheads.s1</td>
<td>Square 1</td>
</tr>
<tr>
<td>noteheads.s2</td>
<td>Square 2</td>
</tr>
</tbody>
</table>

### Special Notehead Glyphs

<table>
<thead>
<tr>
<th>Notehead</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>noteheads.sM1double</td>
<td>Square medium 1 double</td>
</tr>
<tr>
<td>noteheads.s0diamond</td>
<td>Square 0 diamond</td>
</tr>
<tr>
<td>noteheads.s1diamond</td>
<td>Square 1 diamond</td>
</tr>
<tr>
<td>noteheads.s2diamond</td>
<td>Square 2 diamond</td>
</tr>
<tr>
<td>noteheads.s0triangle</td>
<td>Square 0 triangle</td>
</tr>
<tr>
<td>noteheads.d1triangle</td>
<td>Downright 1 triangle</td>
</tr>
<tr>
<td>noteheads.u1triangle</td>
<td>Upright 1 triangle</td>
</tr>
<tr>
<td>noteheads.u2triangle</td>
<td>Upright 2 triangle</td>
</tr>
<tr>
<td>noteheads.d2triangle</td>
<td>Downright 2 triangle</td>
</tr>
<tr>
<td>noteheads.s0slash</td>
<td>Square 0 slash</td>
</tr>
</tbody>
</table>

---

Default Notehead glyphs

<table>
<thead>
<tr>
<th>Notehead</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>noteheads.uM2</td>
<td>Upright medium 2</td>
</tr>
<tr>
<td>noteheads.dM2</td>
<td>Downright medium 2</td>
</tr>
<tr>
<td>noteheads.sM1</td>
<td>Square medium 1</td>
</tr>
<tr>
<td>noteheads.s0</td>
<td>Square 0</td>
</tr>
<tr>
<td>noteheads.s1</td>
<td>Square 1</td>
</tr>
<tr>
<td>noteheads.s2</td>
<td>Square 2</td>
</tr>
</tbody>
</table>

Special Notehead glyphs

<table>
<thead>
<tr>
<th>Notehead</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>noteheads.sM1double</td>
<td>Square medium 1 double</td>
</tr>
<tr>
<td>noteheads.s0diamond</td>
<td>Square 0 diamond</td>
</tr>
<tr>
<td>noteheads.s1diamond</td>
<td>Square 1 diamond</td>
</tr>
<tr>
<td>noteheads.s2diamond</td>
<td>Square 2 diamond</td>
</tr>
<tr>
<td>noteheads.s0triangle</td>
<td>Square 0 triangle</td>
</tr>
<tr>
<td>noteheads.d1triangle</td>
<td>Downright 1 triangle</td>
</tr>
<tr>
<td>noteheads.u1triangle</td>
<td>Upright 1 triangle</td>
</tr>
<tr>
<td>noteheads.u2triangle</td>
<td>Upright 2 triangle</td>
</tr>
<tr>
<td>noteheads.d2triangle</td>
<td>Downright 2 triangle</td>
</tr>
<tr>
<td>noteheads.s0slash</td>
<td>Square 0 slash</td>
</tr>
</tbody>
</table>
### Shape-note Notehead glyphs

<table>
<thead>
<tr>
<th>Notehead</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>noteheads.s0do</td>
<td></td>
</tr>
<tr>
<td>noteheads.u1do</td>
<td></td>
</tr>
<tr>
<td>noteheads.u2do</td>
<td></td>
</tr>
<tr>
<td>noteheads.d1do</td>
<td></td>
</tr>
<tr>
<td>noteheads.d2do</td>
<td></td>
</tr>
<tr>
<td>noteheads.s0doThin</td>
<td></td>
</tr>
<tr>
<td>noteheads.u1doThin</td>
<td></td>
</tr>
<tr>
<td>noteheads.d1doThin</td>
<td></td>
</tr>
<tr>
<td>noteheads.d2doThin</td>
<td></td>
</tr>
<tr>
<td>noteheads.s0re</td>
<td></td>
</tr>
<tr>
<td>noteheads.u1re</td>
<td></td>
</tr>
<tr>
<td>noteheads.d1re</td>
<td></td>
</tr>
<tr>
<td>noteheads.d2re</td>
<td></td>
</tr>
<tr>
<td>noteheads.s0reThin</td>
<td></td>
</tr>
<tr>
<td>noteheads.u1reThin</td>
<td></td>
</tr>
<tr>
<td>noteheads.d1reThin</td>
<td></td>
</tr>
<tr>
<td>noteheads.d2reThin</td>
<td></td>
</tr>
<tr>
<td>noteheads.s0mi</td>
<td></td>
</tr>
<tr>
<td>noteheads.s1mi</td>
<td></td>
</tr>
<tr>
<td>notation</td>
<td>notation</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>noteheads.s2mi</td>
<td>noteheads.s0miMirror</td>
</tr>
<tr>
<td>noteheads.s1miMirror</td>
<td>noteheads.s2miMirror</td>
</tr>
<tr>
<td>noteheads.s0miThin</td>
<td>noteheads.s1miThin</td>
</tr>
<tr>
<td>noteheads.s2miThin</td>
<td>noteheads.u0fa</td>
</tr>
<tr>
<td>noteheads.d0fa</td>
<td>noteheads.ulfa</td>
</tr>
<tr>
<td>noteheads.d1fa</td>
<td>noteheads.u2fa</td>
</tr>
<tr>
<td>noteheads.d2fa</td>
<td>noteheads.u0faThin</td>
</tr>
<tr>
<td>noteheads.d0faThin</td>
<td>noteheads.u1faThin</td>
</tr>
<tr>
<td>noteheads.d1faThin</td>
<td>noteheads.u2faThin</td>
</tr>
<tr>
<td>noteheads.d2faThin</td>
<td>noteheads.s0sol</td>
</tr>
<tr>
<td>noteheads.s1sol</td>
<td>noteheads.s2sol</td>
</tr>
<tr>
<td>noteheads.s0la</td>
<td>noteheads.s1la</td>
</tr>
<tr>
<td>noteheads.s2la</td>
<td>noteheads.s0laThin</td>
</tr>
<tr>
<td>noteheads.s1laThin</td>
<td>noteheads.s2laThin</td>
</tr>
<tr>
<td>noteheads.s0ti</td>
<td>noteheads.ulti</td>
</tr>
</tbody>
</table>
Appendix A: Notation manual tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Column 1</th>
<th>Column 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>noteheads.d1ti</td>
<td>noteheads.u2ti</td>
<td></td>
</tr>
<tr>
<td>noteheads.d2ti</td>
<td>noteheads.s0tiThin</td>
<td></td>
</tr>
<tr>
<td>noteheads.ultiThin</td>
<td>noteheads.d1tiThin</td>
<td></td>
</tr>
<tr>
<td>noteheads.u2tiThin</td>
<td>noteheads.d2tiThin</td>
<td></td>
</tr>
<tr>
<td>noteheads.u0doFunk</td>
<td>noteheads.d0doFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.u1doFunk</td>
<td>noteheads.d1doFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.u2doFunk</td>
<td>noteheads.d2doFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.u0reFunk</td>
<td>noteheads.d0reFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.u1reFunk</td>
<td>noteheads.d1reFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.u2reFunk</td>
<td>noteheads.d2reFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.u0miFunk</td>
<td>noteheads.d0miFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.u1miFunk</td>
<td>noteheads.d1miFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.s2miFunk</td>
<td>noteheads.u0faFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.d0faFunk</td>
<td>noteheads.u1faFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.d1faFunk</td>
<td>noteheads.u2faFunk</td>
<td></td>
</tr>
<tr>
<td>Notation</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>noteheads.d2faFunk</td>
<td>noteheads.s0solFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.s1solFunk</td>
<td>noteheads.s2solFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.s0laFunk</td>
<td>noteheads.s1laFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.s2laFunk</td>
<td>noteheads.u0tiFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.d0tiFunk</td>
<td>noteheads.ultiFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.d1tiFunk</td>
<td>noteheads.u2tiFunk</td>
<td></td>
</tr>
<tr>
<td>noteheads.d2tiFunk</td>
<td>noteheads.s0doWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.u1doWalker</td>
<td>noteheads.d1doWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.u2doWalker</td>
<td>noteheads.d2doWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.s0reWalker</td>
<td>noteheads.u1reWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.d1reWalker</td>
<td>noteheads.u2reWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.d2reWalker</td>
<td>noteheads.s0miWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.s1miWalker</td>
<td>noteheads.s2miWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.s0faWalker</td>
<td>noteheads.u1faWalker</td>
<td></td>
</tr>
<tr>
<td>noteheads.d1faWalker</td>
<td>noteheads.u2faWalker</td>
<td></td>
</tr>
</tbody>
</table>
Appendix A: Notation manual tables

---

**Noteheads**

- 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.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
m m p
r r s
z z
## Script glyphs

<table>
<thead>
<tr>
<th>Script Glyphs</th>
<th>Unicode</th>
</tr>
</thead>
<tbody>
<tr>
<td>scripts.ufermata</td>
<td>☺</td>
</tr>
<tr>
<td>scripts.dfermata</td>
<td>☻</td>
</tr>
<tr>
<td>scripts.ushortfermata</td>
<td>∧</td>
</tr>
<tr>
<td>scripts.dshortfermata</td>
<td>∨</td>
</tr>
<tr>
<td>scripts.ulongfermata</td>
<td>□</td>
</tr>
<tr>
<td>scripts.dlongfermata</td>
<td>▼</td>
</tr>
<tr>
<td>scripts.uverylongfermata</td>
<td>△</td>
</tr>
<tr>
<td>scripts.dverylongfermata</td>
<td>▴</td>
</tr>
<tr>
<td>scripts.thumb</td>
<td>Ø</td>
</tr>
<tr>
<td>scripts.sforzato</td>
<td>&gt;</td>
</tr>
<tr>
<td>scripts.espr</td>
<td>&lt;&gt;</td>
</tr>
<tr>
<td>scripts.staccato</td>
<td>.</td>
</tr>
<tr>
<td>scripts.ustaccatissimo</td>
<td>′</td>
</tr>
<tr>
<td>scripts.dstaccatissimo</td>
<td>‴</td>
</tr>
<tr>
<td>scripts.tenuto</td>
<td>_</td>
</tr>
<tr>
<td>scripts.uportato</td>
<td>\</td>
</tr>
<tr>
<td>scripts.dportato</td>
<td>.</td>
</tr>
<tr>
<td>scripts.umarcato</td>
<td>^</td>
</tr>
<tr>
<td>scripts.dmarcato</td>
<td>▽</td>
</tr>
<tr>
<td>scripts.open</td>
<td>o</td>
</tr>
<tr>
<td>scripts.halfopen</td>
<td>$</td>
</tr>
<tr>
<td>scripts.halfopenvertical</td>
<td>$</td>
</tr>
<tr>
<td>scripts.stopped</td>
<td>+</td>
</tr>
<tr>
<td>scripts.upbow</td>
<td>\</td>
</tr>
<tr>
<td>scripts.reverseturn</td>
<td>$</td>
</tr>
<tr>
<td>scripts.turn</td>
<td>∞</td>
</tr>
<tr>
<td>scripts.trill</td>
<td>tr</td>
</tr>
</tbody>
</table>
Appendix A: Notation manual tables

\[ \text{scripts.upedalheel} \cup \text{scripts.dpedalheel} \cap \]
\[ \text{scripts.upedaltoe} \lor \text{scripts.dpedaltoe} \land \]
\[ \text{scripts.flageolet} \circ \text{scripts.segno} \cdot \]
\[ \text{scripts.varsegno} \circ \text{scripts.coda} \cdot \]
\[ \text{scripts.varcoda} \oplus \text{scripts.rcomma} \]
\[ \text{scripts.lcomma} \ominus \text{scripts.rvarcomma} \
\]
\[ \text{scripts.lvarcomma} \ominus \text{scripts.arpeggio} \]
\[ \text{scripts.trill_element} \sim \text{scripts.arpeggio} \cdot \text{arrow.M1} \]
\[ \text{scripts.arpeggio.arrow.1} \rightharpoonup \text{scripts.trilelement} \]
\[ \text{scripts.prall} \leftharpoonup \text{scripts.mordent} \]
\[ \text{scripts.prallprall} \leftharpoonup \text{scripts.prallmordent} \]
\[ \text{scripts.upprall} \leftharpoonup \text{scripts.upmordent} \]
\[ \text{scripts.pralldown} \leftharpoonup \text{scripts.downprall} \]
\[ \text{scripts.downmordent} \leftharpoonup \text{scripts.prallup} \]
\[ \text{scripts.lineprall} \leftharpoonup \text{scripts.caesura.curved} \]}
### Appendix A: Notation manual tables

<table>
<thead>
<tr>
<th>Script Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>scripts.caesura.straight</code></td>
<td>Caesura straight</td>
</tr>
<tr>
<td><code>scripts.snappizzicato</code></td>
<td>Snappizzicato</td>
</tr>
<tr>
<td><code>scripts.uaccentus</code></td>
<td>Uaccentus</td>
</tr>
<tr>
<td><code>scripts.usemicirculus</code></td>
<td>Usemicirculus</td>
</tr>
<tr>
<td><code>scripts.circulus</code></td>
<td>Circulus</td>
</tr>
<tr>
<td><code>scripts.usignumcongruentiae</code></td>
<td>Usignum congruentiae</td>
</tr>
</tbody>
</table>

#### Arrowhead glyphs

<table>
<thead>
<tr>
<th>Arrowhead Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>arrowheads.open.01</code></td>
<td>Open arrowhead</td>
</tr>
<tr>
<td><code>arrowheads.open.0M1</code></td>
<td>Open arrowhead</td>
</tr>
<tr>
<td><code>arrowheads.open.11</code></td>
<td>Open arrowhead</td>
</tr>
<tr>
<td><code>arrowheads.open.1M1</code></td>
<td>Open arrowhead</td>
</tr>
<tr>
<td><code>arrowheads.close.01</code></td>
<td>Close arrowhead</td>
</tr>
<tr>
<td><code>arrowheads.close.0M1</code></td>
<td>Close arrowhead</td>
</tr>
<tr>
<td><code>arrowheads.close.11</code></td>
<td>Close arrowhead</td>
</tr>
<tr>
<td><code>arrowheads.close.1M1</code></td>
<td>Close arrowhead</td>
</tr>
</tbody>
</table>

#### Bracket-tip glyphs

<table>
<thead>
<tr>
<th>Brackettip Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>brackettips.up</code></td>
<td>Brackettip up</td>
</tr>
<tr>
<td><code>brackettips.down</code></td>
<td>Brackettip down</td>
</tr>
</tbody>
</table>

#### Pedal glyphs

<table>
<thead>
<tr>
<th>Pedal Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pedal.*</code></td>
<td>Pedal</td>
</tr>
<tr>
<td><code>pedal..</code></td>
<td>Pedal</td>
</tr>
<tr>
<td><code>pedal.d</code></td>
<td>Pedal</td>
</tr>
<tr>
<td><code>pedal.Ped</code></td>
<td>Pedal</td>
</tr>
</tbody>
</table>

---
Accordion glyphs

accordion.discant\[accordion.discant\]
\[accordion.dot\]

accordion.freebass\[accordion.freebass\]
\[accordion.stdbass\]

accordion.bayanbass\[accordion.bayanbass\]
\[accordion.oldEE\]

accordion.push\[accordion.push\]
\[accordion.pull\]

Tie glyphs

ties.lyric.short\[ties.lyric.short\]
\[ties.lyric.default\]

Vaticana glyphs

clefs.vaticana.do\[clefs.vaticana.do\]
\[clefs.vaticana.do_change\]

clefs.vaticana.fa\[clefs.vaticana.fa\]
\[clefs.vaticana.fa_change\]

custodes.vaticana.u0\[custodes.vaticana.u0\]
\[custodes.vaticana.u1\]

custodes.vaticana.u2\[custodes.vaticana.u2\]
\[custodes.vaticana.d0\]

custodes.vaticana.d1\[custodes.vaticana.d1\]
\[custodes.vaticana.d2\]

accidentals.vaticanaM1\[accidentals.vaticanaM1\]
\[accidentals.vaticana0\]

dots.dotvaticana\[dots.dotvaticana\]
\[noteheads.svaticana.punctum\]

noteheads.svaticana\[noteheads.svaticana\]
\[.linea.punctum\]
<table>
<thead>
<tr>
<th>Noteheads</th>
<th>Medicaea Glyphs</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>noteheads.svaticana.linea.punctum.cavum</code></td>
<td><code>noteheads.svaticana.inclinatum</code></td>
</tr>
<tr>
<td><code>noteheads.svaticana.lpes</code></td>
<td><code>noteheads.svaticana.vlpes</code></td>
</tr>
<tr>
<td><code>noteheads.svaticana.upes</code></td>
<td><code>noteheads.svaticana.vupes</code></td>
</tr>
<tr>
<td><code>noteheads.svaticana.plica</code></td>
<td><code>noteheads.svaticana.vplica</code></td>
</tr>
<tr>
<td><code>noteheads.svaticana.epiphonus</code></td>
<td><code>noteheads.svaticana.vepiphonus</code></td>
</tr>
<tr>
<td><code>noteheads.svaticana.reverse.plica</code></td>
<td><code>noteheads.svaticana.reverse.vplica</code></td>
</tr>
<tr>
<td><code>noteheads.svaticana.inner.cephalicus</code></td>
<td><code>noteheads.svaticana.cephalicus</code></td>
</tr>
</tbody>
</table>

**Medicaea Glyphs**

<table>
<thead>
<tr>
<th>Clefs</th>
<th>Custodes</th>
<th>Accidentals</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>clefs.medicaea.do</code></td>
<td><code>clefs.medicaea.do_change</code></td>
<td><code>noteheads.smedicaea.inclinatum</code></td>
</tr>
<tr>
<td><code>clefs.medicaea.fa</code></td>
<td><code>clefs.medicaea.fa_change</code></td>
<td><code>noteheads.smedicaea.rvirga</code></td>
</tr>
<tr>
<td><code>custodes.medicaea.u0</code></td>
<td><code>custodes.medicaea.u1</code></td>
<td><code>noteheads.smedicaea.punctum</code></td>
</tr>
<tr>
<td><code>custodes.medicaea.u2</code></td>
<td><code>custodes.medicaea.d0</code></td>
<td><code>noteheads.smedicaea.virga</code></td>
</tr>
<tr>
<td><code>custodes.medicaea.d1</code></td>
<td><code>custodes.medicaea.d2</code></td>
<td><code>noteheads.smedicaea.quilisma</code></td>
</tr>
<tr>
<td><code>accidentals.medicaeaM1</code></td>
<td><code>noteheads.smedicaea.inclinatum</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>noteheads.smedicaea.rvirga</code></td>
<td></td>
</tr>
</tbody>
</table>
## Hufnagel glyphs

<table>
<thead>
<tr>
<th>Glyphs</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>clefs.hufnagel.do</td>
<td>r</td>
</tr>
<tr>
<td>clefs.hufnagel.do_change</td>
<td>r</td>
</tr>
<tr>
<td>clefs.hufnagel.fa</td>
<td>f</td>
</tr>
<tr>
<td>clefs.hufnagel.fa_change</td>
<td>f</td>
</tr>
<tr>
<td>clefs.hufnagel.do.fa</td>
<td>r</td>
</tr>
<tr>
<td>clefs.hufnagel.do_fa_change</td>
<td>f</td>
</tr>
<tr>
<td>custodes.hufnagel.u0</td>
<td>✓</td>
</tr>
<tr>
<td>custodes.hufnagel.u1</td>
<td>✓</td>
</tr>
<tr>
<td>custodes.hufnagel.u2</td>
<td>✓</td>
</tr>
<tr>
<td>custodes.hufnagel.d0</td>
<td>✓</td>
</tr>
<tr>
<td>custodes.hufnagel.d1</td>
<td>✓</td>
</tr>
<tr>
<td>custodes.hufnagel.d2</td>
<td>✓</td>
</tr>
<tr>
<td>accidentals.hufnagelM1</td>
<td>❌</td>
</tr>
<tr>
<td>noteheads.shufnagel.punctum</td>
<td>❌</td>
</tr>
<tr>
<td>noteheads.shufnagel.virga</td>
<td>❌</td>
</tr>
<tr>
<td>noteheads.shufnagel.lpes</td>
<td>❌</td>
</tr>
</tbody>
</table>

## Mensural glyphs

<table>
<thead>
<tr>
<th>Glyphs</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>rests.M3mensural</td>
<td></td>
</tr>
<tr>
<td>rests.M2mensural</td>
<td></td>
</tr>
<tr>
<td>rests.M1mensural</td>
<td>,</td>
</tr>
<tr>
<td>rests.0mensural</td>
<td></td>
</tr>
<tr>
<td>rests.1mensural</td>
<td></td>
</tr>
<tr>
<td>rests.2mensural</td>
<td></td>
</tr>
<tr>
<td>rests.3mensural</td>
<td></td>
</tr>
<tr>
<td>rests.4mensural</td>
<td></td>
</tr>
<tr>
<td>clefs.mensural.c</td>
<td>H</td>
</tr>
<tr>
<td>clefs.mensural.c_change</td>
<td>H</td>
</tr>
<tr>
<td>clefs.blackmensural.c</td>
<td>H</td>
</tr>
<tr>
<td>clefs.blackmensural.c_change</td>
<td>H</td>
</tr>
</tbody>
</table>
Appendix A: Notation manual tables

\begin{verbatim}
clefs.mensural.f  \x;  clefs.mensural.f_change  \x;  
clefs.mensural.g  \g;  clefs.mensural.g_change  \g;  
custodes.mensural.u0  \u/;  custodes.mensural.u1  \u/;  
custodes.mensural.u2  \u/;  custodes.mensural.d0  \u/;  
custodes.mensural.d1  \u/;  custodes.mensural.d2  \u/;  
accidentals.mensural1  \x;  accidentals.mensuralM1  \b;  
flags.mensuralu03  \u/;  flags.mensuralu13  \u/;  
flags.mensuralu23  \u/;  flags.mensurald03  \u/;  
flags.mensurald13  \u/;  flags.mensural23  \u/;  
flags.mensuralu04  \u/;  flags.mensuralu14  \u/;  
flags.mensuralu24  \u/;  flags.mensurald04  \u/;  
flags.mensurald14  \u/;  flags.mensurald24  \u/;  
flags.mensuralu05  \u/;  flags.mensuralu15  \u/;  
flags.mensuralu25  \u/;  flags.mensurald05  \u/;  
flags.mensurald15  \u/;  flags.mensurald25  \u/;  
\end{verbatim}
flags.mensural06 \rightarrow flags.mensural16
flags.mensural26 \rightarrow flags.mensural06
flags.mensural16 \rightarrow flags.mensural26

timesig.mensural44 \rightarrow timesig.mensural22

\begin{align*}
timesig.mensural32 & \overset{O}{\rightarrow} \quad \text{timesig.mensural64} & \overset{C}{\rightarrow} \\
timesig.mensural94 & \overset{O}{\rightarrow} \quad \text{timesig.mensural34} & \overset{\phi}{\rightarrow} \\
timesig.mensural68 & \overset{\phi}{\rightarrow} \quad \text{timesig.mensural98} & \overset{\phi}{\rightarrow} \\
timesig.mensural48 & \overset{\circ}{\rightarrow} \quad \text{timesig.mensural68alt} & \overset{\varnothing}{\rightarrow} \\
timesig.mensural24 & \overset{\varnothing}{\rightarrow} \quad \text{noteheads.uM3mensural} & \overset{\Pi}{\rightarrow} \\
\text{noteheads.dM3mensural} & \overset{\Pi}{\rightarrow} \quad \text{noteheads.sM3ligmensural} & \overset{\Pi}{\rightarrow} \\
\text{noteheads.uM2mensural} & \overset{\Pi}{\rightarrow} \quad \text{noteheads.dM2mensural} & \overset{\Pi}{\rightarrow} \\
\text{noteheads.sM2ligmensural} & \overset{\Pi}{\rightarrow} \quad \text{noteheads.sM1mensural} & \overset{\Pi}{\rightarrow} \\
\text{noteheads.urM3mensural} & \overset{\Pi}{\rightarrow} \quad \text{noteheads.drM3mensural} & \overset{\Pi}{\rightarrow} \\
\text{noteheads.srM3ligmensural} & \overset{\Pi}{\rightarrow} \quad \text{noteheads.urM2mensural} & \overset{\Pi}{\rightarrow} \\
\text{noteheads.drM2mensural} & \overset{\Pi}{\rightarrow} \quad \text{noteheads.srM2ligmensural} & \overset{\Pi}{\rightarrow} \
\end{align*}
<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>noteheads.srM1mensural</td>
<td>noteheads.uM3semimensural</td>
</tr>
<tr>
<td>noteheads .dM3semimensural</td>
<td>noteheads .sM3semiligmensural</td>
</tr>
<tr>
<td>noteheads .uM2semimensural</td>
<td>noteheads .dM2semimensural</td>
</tr>
<tr>
<td>noteheads .sM2semiligmensural</td>
<td>noteheads .sM1semimensural</td>
</tr>
<tr>
<td>noteheads .urM3semimensural</td>
<td>noteheads .drM3semimensural</td>
</tr>
<tr>
<td>noteheads .srM3semiligmensural</td>
<td>noteheads .urM2semimensural</td>
</tr>
<tr>
<td>noteheads .drM2semimensural</td>
<td>noteheads .srM2semiligmensural</td>
</tr>
<tr>
<td>noteheads .srM1semimensural</td>
<td>noteheads .uM3blackmensural</td>
</tr>
<tr>
<td>noteheads .dM3blackmensural</td>
<td>noteheads .sM3blackligmensural</td>
</tr>
<tr>
<td>noteheads .uM2blackmensural</td>
<td>noteheads .dM2blackmensural</td>
</tr>
<tr>
<td>noteheads .sM2blackligmensural</td>
<td>noteheads .sM1blackmensural</td>
</tr>
<tr>
<td>noteheads.s0mensural</td>
<td>noteheads.s1mensural</td>
</tr>
<tr>
<td>noteheads.s2mensural</td>
<td>noteheads.s0blackmensural</td>
</tr>
</tbody>
</table>
### Neomensural glyphs

<table>
<thead>
<tr>
<th>Neomensural Glyphs</th>
<th>Neemensural Glyphs</th>
<th>Neomensural Glyphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>rests.M3neomensural</td>
<td>II</td>
<td>rests.M2neomensural</td>
</tr>
<tr>
<td>rests.M1neomensural</td>
<td>I</td>
<td>rests.0neomensural</td>
</tr>
<tr>
<td>rests.1neomensural</td>
<td>.</td>
<td>rests.2neomensural</td>
</tr>
<tr>
<td>rests.3neomensural</td>
<td>.</td>
<td>rests.4neomensural</td>
</tr>
<tr>
<td>clefs.neomensural.c</td>
<td>H</td>
<td>clefs.neomensural.c_change</td>
</tr>
<tr>
<td>timesig.neomensural44</td>
<td>C</td>
<td>timesig.neomensural22</td>
</tr>
<tr>
<td>timesig.neomensural32</td>
<td>O</td>
<td>timesig.neomensural64</td>
</tr>
<tr>
<td>timesig.neomensural94</td>
<td>O</td>
<td>timesig.neomensural34</td>
</tr>
<tr>
<td>timesig.neomensural68</td>
<td>C</td>
<td>timesig.neomensural98</td>
</tr>
<tr>
<td>timesig.neomensural48</td>
<td>C</td>
<td>timesig.neomensural68alt</td>
</tr>
<tr>
<td>timesig.neomensural24</td>
<td>O</td>
<td>noteheads.uM3neomensural</td>
</tr>
<tr>
<td>noteheads.dM3neomensural</td>
<td>II</td>
<td>noteheads.uM2neomensural</td>
</tr>
<tr>
<td>noteheads.dM2neomensural</td>
<td>II</td>
<td>noteheads.sM1neomensural</td>
</tr>
<tr>
<td>noteheads.urM3neomensural</td>
<td>II</td>
<td>noteheads.drM3neomensural</td>
</tr>
</tbody>
</table>
### Petrucci glyphs

<table>
<thead>
<tr>
<th>Clef</th>
<th>Petrucci</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>clefs.petrucci.c1</td>
<td>clefs.petrucci.c1_change</td>
<td></td>
</tr>
<tr>
<td>clefs.petrucci.c2</td>
<td>clefs.petrucci.c2_change</td>
<td></td>
</tr>
<tr>
<td>clefs.petrucci.c3</td>
<td>clefs.petrucci.c3_change</td>
<td></td>
</tr>
<tr>
<td>clefs.petrucci.c4</td>
<td>clefs.petrucci.c4_change</td>
<td></td>
</tr>
<tr>
<td>clefs.petrucci.c5</td>
<td>clefs.petrucci.c5_change</td>
<td></td>
</tr>
<tr>
<td>clefs.petrucci.f</td>
<td>clefs.petrucci.f_change</td>
<td></td>
</tr>
<tr>
<td>clefs.petrucci.g</td>
<td>clefs.petrucci.g_change</td>
<td></td>
</tr>
</tbody>
</table>

### Notation manual tables

- `noteheads.urM2neomensural`
- `noteheads.drM2neomensural`
- `noteheads.srM1neomensural`
- `noteheads.s0neomensural`
- `noteheads.s1neomensural`
- `noteheads.s2neomensural`
- `noteheads.s0petrucci`
- `noteheads.s1petrucci`
- `noteheads.s2petrucci`
- `noteheads.s0blackpetrucci`
- `noteheads.s1blackpetrucci`
- `noteheads.s2blackpetrucci`
**Solesmes glyphs**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>noteheads.ssolesmes</code></td>
<td></td>
</tr>
<tr>
<td><code>.incl.parvum</code></td>
<td></td>
</tr>
<tr>
<td><code>noteheads.ssolesmes.auct.asc</code></td>
<td></td>
</tr>
<tr>
<td><code>noteheads.ssolesmes.auct.desc</code></td>
<td></td>
</tr>
<tr>
<td><code>noteheads.ssolesmes.stropha</code></td>
<td></td>
</tr>
<tr>
<td><code>.stropha.aucta</code></td>
<td></td>
</tr>
<tr>
<td><code>noteheads.ssolesmes.oriscus</code></td>
<td></td>
</tr>
</tbody>
</table>

**Kievan Notation glyphs**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>clefs.kievan.do</code></td>
<td></td>
</tr>
<tr>
<td><code>clefs.kievan.do_change</code></td>
<td></td>
</tr>
<tr>
<td><code>accidentals.kievan1</code></td>
<td></td>
</tr>
<tr>
<td><code>accidentals.kievanM1</code></td>
<td></td>
</tr>
<tr>
<td><code>scripts.barline.kievan</code></td>
<td></td>
</tr>
<tr>
<td><code>dots.dotkievan</code></td>
<td></td>
</tr>
<tr>
<td><code>noteheads.sM2kievan</code></td>
<td></td>
</tr>
<tr>
<td><code>noteheads.sM1kievan</code></td>
<td></td>
</tr>
<tr>
<td><code>noteheads.s0kievan</code></td>
<td></td>
</tr>
<tr>
<td><code>noteheads.d2kievan</code></td>
<td></td>
</tr>
<tr>
<td><code>noteheads.u2kievan</code></td>
<td></td>
</tr>
<tr>
<td><code>noteheads.s1kievan</code></td>
<td></td>
</tr>
<tr>
<td><code>noteheads.srlkievan</code></td>
<td></td>
</tr>
<tr>
<td><code>noteheads.d3kievan</code></td>
<td></td>
</tr>
<tr>
<td><code>noteheads.u3kievan</code></td>
<td></td>
</tr>
</tbody>
</table>
A.9 Note head styles

The following styles may be used for note heads.

A.10 Text markup commands

The following commands can all be used inside \markup { }.

A.10.1 Font

\abs-fontsize size (number) arg (markup)

Use size as the absolute font size to display arg. Adjusts baseline-skip and word-space accordingly.
\markup{
  \default \text font size
  \hspace #2
  \text font size 16
  \hspace #2
  \text font size 12
}

default text font size text font size 16 text font size 12

\bold (markup)
Switch to bold font-series.
\markup{
  \default
  \hspace #2
  \bold
}

bold

\box (markup)
Draw a box around \arg. Looks at \textit{thickness}, \textit{box-padding} and \textit{font-size} properties to determine line thickness and padding around the markup.
\markup{
  \override #'(box-padding . 0.5)
  \box
  \line \{ V. S. \}
}

\text\v S.

Used properties:
- box-padding (0.2)
- font-size (0)
- thickness (1)

\caps (markup)
Copy of the \textit{\smallCaps} command.
\markup{
  \default
  \hspace #2
  \caps
    \text { Text in small caps}
}

default TEXT IN SMALL CAPS

\dynamic (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
}

\fontsize increment (number) arg (markup)
Add increment to the font-size. Adjusts baseline-skip accordingly.

\huge arg (markup)
Set font size to +2.
\markup { 
  \default 
  \hspace #2
  \huge
}

\huge arg (markup)
Set font size to +2.
\markup { 
  \default 
  \hspace #2
  \huge
}

\default \huge

\italic arg (markup)
Use italic font-shape for arg.
Appendix A: Notation manual tables

\[ \text{\textbackslash{markup}} \{ \]
\text{default}
\text{\textbackslash{hspace}} #2
\text{\textbackslash{italic}}
\text{italic}
\}

\text{default italic}

\text{\textbackslash{large}} \text{arg} \text{(markup)}
Set font size to +1.
\[ \text{\textbackslash{markup}} \{ \]
\text{default}
\text{\textbackslash{hspace}} #2
\text{\textbackslash{large}}
\text{large}
\}

\text{default large}

\text{\textbackslash{larger}} \text{arg} \text{(markup)}
Increase the font size relative to the current setting.
\[ \text{\textbackslash{markup}} \{ \]
\text{default}
\text{\textbackslash{hspace}} #2
\text{\textbackslash{larger}}
\text{larger}
\}

\text{default larger}

\text{\textbackslash{magnify}} \text{sz} \text{(number)} \text{arg} \text{(markup)}
Set the font magnification for its argument. In the following example, the middle A is 10% larger:
\text{A \textbackslash{magnify} #1.1 \{ A \} A}
Note: Magnification only works if a font name is explicitly selected. Use \textbackslash{fontsize} otherwise.
\[ \text{\textbackslash{markup}} \{ \]
\text{default}
\text{\textbackslash{hspace}} #2
\text{\textbackslash{magnify}} #1.5 \{
\text{ 50\% larger}
\}
\}

\text{default 50\% larger}

\text{\textbackslash{medium}} \text{arg} \text{(markup)}
Switch to medium font-series (in contrast to bold).
Appendix A: Notation manual tables

\markup {
  \bold {
    some bold text
    \hspace #2
    \medium {
      medium font series
    }
    \hspace #2
    bold again
  }
}

\textbf{some bold text}  medium font series  \textbf{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:
• baseline-skip

\normal-size-super arg (markup)
Set arg in superscript with a normal font size.
\markup {
  default
  \normal-size-super {
    superscript in standard size
  }
}

superscript in standard size default

Used properties:
• baseline-skip

\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
    }
  }
}
Appendix A: Notation manual tables

\newcommand{\hspace}{#2}
as before
}
}

HUGE BOLD SANS CAPS  huge normal  AS BEFORE

\normalsize{arg} (markup)
Set font size to default.
\markup{
  \textit{this is very small}
  \hspace{#2}
  \normalsize{
    normal size
  }
  \hspace{#2}
  \textit{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{
  \textit{0 1 2 3 4 5 6 7 8 9 . ,}
}

0123456789.,

\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{#1}{("thx" . "Thanks!")}{thx}}
Thanks!

\roman{arg} (markup)
Set font family to roman.
\markup{
  \textit{sans \textbf{bold}}{
    sans serif, bold
    \hspace{#2}
    \roman{
      text in roman font family
    }
  }
}
\hspace #2
\returntosans
}
}

\sansserif,\bold\textinromanfontfamily\returntosans

\sans arg (markup)
Switch to the sans serif font family.
\markup{
default
\hspace #2
\sans{
sans serif
}
}

default sans serif

\simplestr (string)
A simple text string; \markup{\foo} is equivalent with \markup{\simplestr\"foo\"}.
Note: for creating standard text markup or defining new markup commands, the use of \simplestr is unnecessary.
\markup{
\simplestr\"simple\"
\simplestr\"text\"
\simplestr\"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{
\smallcaps
Text in small caps
}
Appendix A: Notation manual tables

\smaller{arg} (markup)
Decrease the font size relative to the current setting.
\markup{\fontsize{#3.5}{\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}\O}}

H$_2$O

Used properties:
- baseline-skip
- font-size (0)

\super{arg} (markup)
Set arg in superscript.
\markup{E = \concat{mc\super{2}}}

E = mc$^2$

Used properties:
Appendix A: Notation manual tables

- baseline-skip
- font-size \(0\)

\texttt{\textbackslash teeny arg} (markup)
Set font size to -3.
\begin{verbatim}
\markup {
  \default \hspace #2 \texttt{\textbackslash teeny}
  \texttt{\textbackslash tiny}
  \texttt{\textbackslash typewriter}
}
\end{verbatim}
\begin{verbatim}
default \texttt{\textbackslash teeny}
\end{verbatim}

\texttt{\textbackslash text arg} (markup)
Use a text font instead of music symbol or music alphabet font.
\begin{verbatim}
\markup {
  \default \hspace #2 \texttt{\textbackslash number}
  \texttt{\textbackslash text}
  \texttt{\textbackslash typewriter}
  \texttt{\underline}
}
\end{verbatim}
\begin{verbatim}
default \texttt{\textbackslash text}
\end{verbatim}

\texttt{1, 2, three, four, 5}

\texttt{\textbackslash tiny arg} (markup)
Set font size to -2.
\begin{verbatim}
\markup {
  \default \hspace #2 \texttt{\textbackslash tiny}
  \texttt{\textbackslash typewriter}
  \texttt{\underline}
}
\end{verbatim}
\begin{verbatim}
default \texttt{\textbackslash tiny}
\end{verbatim}

\texttt{\textbackslash typewriter arg} (markup)
Use \texttt{font-family typewriter} for \texttt{arg}.
\begin{verbatim}
\markup {
  \default \hspace #2 \texttt{\textbackslash typewriter}
  \texttt{\underline}
}
\end{verbatim}
\begin{verbatim}
default \texttt{\textbackslash typewriter}
\end{verbatim}

\texttt{\textbackslash underline arg} (markup)
Underline \texttt{arg}. Looks at \texttt{thickness} to determine line thickness, and \texttt{offset} to determine line y-offset.
A.10.2 Align

\center-align arg (markup)
Align arg to its X center.
\markup {\center-align \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
}
}

\text{one}
\text{two}
\text{three}

Used properties:
\begin{itemize}
\item baseline-skip
\end{itemize}

\text{\textbackslash column\text{ args} (markup list)}

Stack the markups in \text{args} vertically. The property \text{baseline-skip} determines the space between markups in \text{args}.

\markup { 
\column { 
one
two
three
}
}

\text{one}
\text{two}
\text{three}

Used properties:
\begin{itemize}
\item baseline-skip
\end{itemize}

\text{\textbackslash combine\text{ arg1} (markup) arg2 (markup)}

Print two markups on top of each other.

Note: \text{\textbackslash combine} cannot take a list of markups enclosed in curly braces as an argument; the follow example will not compile:
\begin{verbatim}
\combine { a list }
\end{verbatim}

\markup { 
\fontsize \#5
\override #'(thickness . 2)
\combine
\draw-line #'(0 . 4)
\arrow-head \text{Y \text{DOWN} \#f}
}

↓

\text{\textbackslash concat\text{ args} (markup list)}

Concatenate \text{args} in a horizontal line, without spaces in between. Strings and simple markups are concatenated on the input level, allowing ligatures. For example, \text{\textbackslash concat\{"f\text{\textbackslash 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 . \DOWN) { 
    \dir-column { 
      going up 
    } 
  } 
}

up up
up
up
up
gong
gong
gong
gong
down
down

gong
gong
gong
gong

Used properties:
\bullet baseline-skip
\bullet 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 } 
      } 
    } 
  } 
}

\\
Appendix A: Notation manual tables

\line \{ across the page \}
\line \{ across the page \}
\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 :

Used properties:
- line-width
- word-space
\general-align \textit{axis} (integer) \textit{dir} (number) \textit{arg} (markup)

Align \textit{arg} in \textit{axis} direction to the \textit{dir} side.

\markup {
\column {
  one
  \general-align #X #LEFT
  two
  three
  \null
  one
  \general-align #X #CENTER
  two
  three
  \null
  \line {
    one
    \general-align #Y #UP
    two
    three
  }
  \null
  \line {
    one
    \general-align #Y #3.2
    two
    three
  }
}
}

one
two
three

one
two
three

one
two
three

one
two
three

two

\halign \textit{dir} (number) \textit{arg} (markup)

Set horizontal alignment. If \textit{dir} is -1, then it is left-aligned, while +1 is right. Values in between interpolate alignment accordingly.

\markup {
\column {
  one
  \halign #LEFT

\center-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 {\center-in #12 Oboe}
  c'\'1
}
\new Staff {
  \set Staff.instrumentName = \markup {\center-in #12 Bassoon}
Appendix A: Notation manual tables

\clef tenor
c'1
}
>>

<table>
<thead>
<tr>
<th>Oboe</th>
<th>Bassoon</th>
</tr>
</thead>
<tbody>
<tr>
<td>\hspace amount (number) Create an invisible object taking up horizontal space amount.</td>
<td></td>
</tr>
<tr>
<td>\markup { one \hspace #2 two \hspace #8 three }</td>
<td></td>
</tr>
</tbody>
</table>

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 adipiscing 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 \texttt{args} (markup list)
Like \texttt{wordwrap}, but with lines stretched to justify the margins. Use \texttt{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 \texttt{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 adipiscing 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
  }
}

\left-column args (markup list)
Put args in a left-aligned column.

\markup {
  \left-column {
    one
    two
    three
  }
}

\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 { 
  \lower #3 
  two 
  three 
}

one three

two

\pad-around amount (number) arg (markup)

Add padding amount all around arg.

\markup { 
  \box { 
    \default 
  } 
  \hspace #2 
  \box { 
    \pad-around #0.5 { 
      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 
    } 
  } 
}
\pad-to-box \textit{x-ext} (pair of numbers) \textit{y-ext} (pair of numbers) \textit{arg} (markup)

Make \textit{arg} take at least \textit{x-ext}, \textit{y-ext} space.

\markup {
\box { default \\
#4 \hspace #4 \box {
\pad-to-box #'(0 . 10) #'(0 . 3) { 
  padded \\
}\}
#}
}

\pad-x \textit{amount} (number) \textit{arg} (markup)

Add padding \textit{amount} around \textit{arg} in the X direction.

\markup {
\box { default \\
#4 \hspace #4 \box {
\pad-x #2 { 
  padded \\
}\}
#}
}

\put-adjacent \textit{axis} (integer) \textit{dir} (direction) \textit{arg1} (markup) \textit{arg2} (markup)

Put \textit{arg2} next to \textit{arg1}, without moving \textit{arg1}.

\raise \textit{amount} (number) \textit{arg} (markup)

Raise \textit{arg} by the distance \textit{amount}. A negative \textit{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.
\begin{itemize}
\item \texttt{baseline-skip}
\item \texttt{hspace #2
\item \texttt{\rotate #45
\item \texttt{\line {rotated 45°
\end{itemize}
\texttt{\textbackslash translate} \texttt{\textbackslash offset} (pair of numbers) \texttt{arg} (markup)

Translate \texttt{arg} relative to its surroundings. \texttt{offset} is a pair of numbers representing the displacement in the X and Y axis.

\texttt{\textbackslash markup \{
* \texttt{\textbackslash translate} #'(2 . 3)
\line \{ translated two spaces right, three up \}
\}}

translated two spaces right, three up

\texttt{\textbackslash translate-scaled} \texttt{\textbackslash offset} (pair of numbers) \texttt{arg} (markup)

Translate \texttt{arg} by \texttt{offset}, scaling the offset by the \texttt{font-size}.

\texttt{\textbackslash markup \{
\texttt{\textbackslash fontsize \#5 \{
* \texttt{\textbackslash translate} #'(2 . 3) \texttt{translate}\}
\texttt{hspace \#2}\}
* \texttt{\textbackslash translate-scaled} #'(2 . 3) \texttt{translate-scaled}\}
\}}

\texttt{translate} \texttt{translate-scaled}

Used properties:
- \texttt{font-size (0)}

\texttt{\textbackslash vcenter} \texttt{arg} (markup)

Align \texttt{arg} to its Y center.

\texttt{\textbackslash markup \{
one \texttt{\textbackslash vcenter}
two three\}}

one two three

\texttt{\textbackslash vspace} \texttt{amount} (number)

Create an invisible object taking up vertical space of \texttt{amount} multiplied by 3.

\texttt{\textbackslash markup \{
\texttt{\textbackslash center-column \{
one \texttt{\textbackslash vspace \#2}
two \texttt{\textbackslash vspace \#5}
three\}}
\}}
\header {
  title = "My title"
  myText = "Lorem ipsum dolor sit amet, consectetur adipiscing 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 adipiscing 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 \(\text{line-width} \cdot X\) to set the line width, where \(X\) is the number of staff spaces.
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.

Used properties:
• text-direction (1)
• word-space
• line-width
• baseline-skip

A.10.3 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
    }
  }
}

▲ Y >>

\beam width (number) slope (number) thickness (number)
Create a beam with the specified parameters.
\markup {
  \beam #5 #1 #2
}

[ ]

\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
  }
}
Appendix A: Notation manual tables

\draw-circle \text{radius} \hspace{1em} \text{thickness} \hspace{1em} \text{filled} (\text{boolean})

A circle of radius \text{radius} and thickness \text{thickness}, optionally filled.

\markup { \draw-circle \text{#2} \hspace{#0.5} \hspace{##f} \\
\hspace{#2} \hspace{##t} \}

\draw-dashed-line \text{dest} (\text{pair of numbers})

A dashed line.

If \text{full-length} is set to \#t (default) the dashed-line extends to the whole length
given by \text{dest}, without white space at beginning or end. \text{off} will then be altered
to fit. To insist on the given (or default) values of \text{on}, \text{off} use \texttt{\override}
\texttt{#'(full-length . \#f)} Manual settings for \text{on}, \text{off} and \text{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) 

}

\draw-dotted-line \text{dest} (\text{pair of numbers})

A dotted line.

The dotted-line always extends to the whole length given by \text{dest}, without white
space at beginning or end. Manual settings for \text{off} are possible to get larger or
smaller space between the dots. The given (or default) value of \text{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)

\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

\epsfile axis (number) size (number) file-name (string)
Inline an EPS image. The image is scaled along axis to size.

\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).

\hbracket arg (markup)
Draw horizontal brackets around arg.
\markup {
  \hbracket {
    \line {
      one two three
    }
  }
}

\oval {arg (markup)}
Draw an oval around arg. Use \texttt{thickness}, \texttt{x-padding}, \texttt{x-padding} and \texttt{font-size} properties to determine line thickness and padding around the markup.

\markup {
  \oval {
    Hi
  }
}

\texttt{Hi}

Used properties:
- \texttt{y-padding} (0.75)
- \texttt{x-padding} (0.75)
- \texttt{font-size} (0)
- \texttt{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
        }
      }
    }
  }
}

\begin{tabular}{|c|c|}
\hline
foo & bah  \\
\hline
bar & baz  \\
\hline
\end{tabular}
Used properties:

- width (0.25)
- thickness (1)
- size (1)
- padding
- angularity (0)

\path \text{thickness} \text{(number)} \text{commands} \text{(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.

\begin{verbatim}
samplePath = #'((moveto 0 0)
             (lineto -1 1)
             (lineto 1 1)
             (lineto 1 -1)
             (curveto -5 -5 -5 5 -1 0)
             (closepath))
\end{verbatim}

\begin{verbatim}
\markup {\path #0.25 #samplePath
\override #'(line-join-style . miter) \path #0.25 #samplePath
\override #'(filled . #t) \path #0.25 #samplePath}
\end{verbatim}

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.

\postscript {#"}
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
"

ringsps = \markup {
  \with-dimensions #'(-0.2 . 1.6) #'(0 . 1.2)
  \postscript #ringsps
}

rings = \relative c'' {
  c2\textbraceleft \rings
  a2_\textbraceright \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 {\textbraceleft Overtura
    \textbraceright}
}

\clefs.G \timesig.C44

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
}
}

\textit{stretched}

\triangle \ \textit{filled} \ (boolean)

A \ triangle, \ either \ filled \ or \ empty.

\markup {
\triangle #\!t \\
\hspace #2 \\
\triangle #\!f
}

\textbf{\triangle}

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
}
}
}

\textit{LilyPond \ ... \ music notation for everyone}

\textbf{A.10.4 Music}

\customTabClef \ num-strings \ (integer) \ staff-space \ (number)

Draw \ a \ tab \ clef \ sans-serif \ style.

\doubleflat

Draw \ a \ double \ flat \ symbol.

\markup {
\doubleflat
}
\doubleslash
\doublesharp

Draw a double sharp symbol.
\markup {\doublesharp}

\fermata

Create a fermata glyph. When \textit{direction} is \texttt{DOWN}, use an inverted glyph. Note that within music, one would usually use the \fermata articulation instead of a markup.
\{c1\markup \fermata d1_\markup \fermata\}

\markup {\fermata \override #`(direction ,DOWN) \fermata}

\natural

Draw a natural symbol.
Construct a note symbol, with stem and flag. By using fractional values for \texttt{dir}, longer or shorter stems can be obtained. Supports all note-head-styles. Supported flag-styles are \texttt{default}, \texttt{old-straight-flag}, \texttt{modern-straight-flag} and \texttt{flat-flag}.

\begin{verbatim}
\markup {
\note-by-number #3 #0 #DOWN
\hspace #2
\note-by-number #1 #2 #0.8
} \hspace #2
\end{verbatim}

\begin{verbatim}
\rest-by-number log (number) dot-count (number) dir (number)
A rest or multi-measure-rest symbol.
\markup {
\rest-by-number #3 #2
\hspace #2
\rest-by-number #0 #1
} \hspace #2
\end{verbatim}

Used properties:
- \texttt{style ('())}
- \texttt{flag-style ('())}
- \texttt{font-size (0)}
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)

\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
    }
  }
}
Appendix A: Notation manual tables

\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
c 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
    }
  }
}

Allegro

\semiflat
\semiflat
\semiflat
\semiflat
\semiflat

Used properties:
- baseline-skip

\semiflat
Draw a semiflat symbol.
\markup {
  \semiflat
}
\markup d
\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\text{ all'onde} algenti Lasciate\text{ in abbandono}"
  \tied-lyric "Impetuosi venti I nostri\text{ affetti sono}"
  \tied-lyric "Ogni diletto\text{ e scoglio} Tutta la vita\text{ 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.10.5 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:

\( \text{(mute } \text{string-number)} \)
Place a small ‘x’ at the top of string \text{string-number}.

\( \text{(open } \text{string-number)} \)
Place a small ‘o’ at the top of string \text{string-number}.

\( \text{(barre } \text{start-string } \text{end-string } \text{fret-number)} \)
Place a barre indicator (much like a tie) from string \text{start-string} to string \text{end-string} at fret \text{fret-number}.

\( \text{(capo } \text{fret-number)} \)
Place a capo indicator (a large solid bar) across the entire fretboard at fret location \text{fret-number}. Also, set fret \text{fret-number} to be the lowest fret on the fret diagram.

\( \text{(place-fret } \text{string-number } \text{fret-number} \text{[finger-value [color-modifier]]]} \)
Place a fret playing indication on string \text{string-number} at fret \text{fret-number} with an optional fingering label \text{finger-value}, and an optional color modifier \text{color-modifier}. By default, the fret playing indicator is a solid dot. This can be globally changed by setting the value of the variable \text{dot-color}. Setting \text{color-modifier} to \text{inverted} inverts the dot color for a specific fingering. If the \text{finger} part of the \text{place-fret} element is present, \text{finger-value} will be displayed according to the setting of the variable \text{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
  #'oobo #'(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.10.6 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, 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

\discant #"1" \discant #"1+0"
\discant #"120" \discant #"131"
\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
\freeBass "1"
\freeBass "11"
\freeBass "10"

\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


\[ \text{\stdBass IV name (string)} \]
\[ \text{\stdBass IV name} \] generates a standard bass accordion register symbol.

To make it available,

\[ \text{#(use-modules (scm accreg))} \]

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
Used properties:

- \font-size (0)

\stdBassV name (string)
\stdBassV name generates a standard bass accordion register symbol.

To make it available,
\(\text{#(use-modules (scm accreg))}\)

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
\stdBassV #"Bass/Alto" \stdBassV #"Soft Bass"
\stdBassV #"Soft Bass/Alto" \stdBassV #"Soft Tenor"
\stdBassV #"Alto" \stdBassV #"Soprano"
\stdBassV #"Tenor" \stdBassV #"Sopranos"
\stdBassV #"Master" \stdBassV #"Solo Bass"

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...
A.10.7 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 }

\footnote
\footnote { mkup (markup) note (markup) } 
Have footnote note act as an annotation to the markup mkup.
\markup { \footnote a b \override #'(padding . 0.2) \footnote c d } 

a c
The footnote will not be annotated automatically.
\fraction
\fraction { arg1 (markup) arg2 (markup) } 
Make a fraction of two markups.
\markup { 
\fraction 355 113 
}

\null
The fraction will not be annotated automatically.

\fromproperty
\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"
\fromproperty { 
\italic 
\fromproperty #'header:myTitle 
}
}
\markup { 
\null 
}

from myTitle
\texttt{\textbackslash left-brace size} (number)
A feta brace in point size \textit{size}.
\begin{verbatim}
\markup {
    \left-brace #35
    \hspace #2
    \left-brace #45
}
\end{verbatim}

\texttt{\lookup glyph-name} (string)
Look up a glyph by name.
\begin{verbatim}
\markup {
    \override #'(font-encoding . fetaBraces) {
        \lookup #"brace200"
        \hspace #2
        \rotate #180
        \lookup #"brace180"
    }
}
\end{verbatim}

\texttt{\markalphabet num} (integer)
Make a markup letter for \textit{num}. The letters start with A to Z and continue with double letters.
\begin{verbatim}
\markup {
    \markalphabet #8
    \hspace #2
    \markalphabet #26
}
\end{verbatim}

\texttt{\markletter num} (integer)
Make a markup letter for \textit{num}. The letters start with A to Z (skipping letter I), and continue with double letters.
\begin{verbatim}
\markup {
    \markletter #8
    \hspace #2
    \markletter #26
}
\end{verbatim}
Appendix A: Notation manual tables

\null
An empty markup with extents of a single point.

\on-the-fly procedure (procedure) arg (markup)
Apply the procedure markup command to arg. procedure should take a single argument.

\override new-prop (pair) arg (markup)
Add the argument new-prop to the property list. Properties may be any property supported by Section “font-interface” in Internals Reference, Section “text-interface” in Internals Reference and Section “instrument-specific-markup-interface” in Internals Reference.

\page-link page-number (number) arg (markup)
Add a link to the page page-number around arg. This only works in the PDF backend.

\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.
\texttt{pattern count (integer) axis (integer) space (number) pattern (markup)}

Prints \textit{count} times a \textit{pattern} markup. Patterns are spaced apart by \textit{space}. Patterns are distributed on \textit{axis}.

```
\markup \column {
  "Horizontally repeated :"
  \pattern #7 #X #2 \flat
  \null
  "Vertically repeated :"
  \pattern #3 #Y #0.5 \flat
}
```

Horizontally repeated :
\begin{verbatim}
\textbackslash b b b b b b b
\end{verbatim}

Vertically repeated :
\begin{verbatim}
\textbackslash b
\textbackslash b
\textbackslash b
\end{verbatim}

\texttt{\textbackslash property-recursive symbol (symbol)}

Print out a warning when a header field markup contains some recursive markup definition.

\texttt{\textbackslash right-brace size (number)}

A feta brace in point size \textit{size}, rotated 180 degrees.

```
\markup {
  \right-brace #45
  \hspace #2
  \right-brace #35
}
```

\begin{verbatim}
\{ \{ \\
\end{verbatim}

\texttt{\textbackslash 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
}
```

\begin{verbatim}
5 7
\end{verbatim}

Used properties:
\begin{itemize}
  \item thickness (1.6)
  \item font-size (0)
\end{itemize}
\stencil \stil \textit{(stencil)}
Use a stencil as markup.
\markup {
  \stencil #(make-circle-stencil 2 0 \texttt{t})
}

\strut
Create a box of the same height as the space in the current font.

\transparent \texttt{arg (markup)}
Make \texttt{arg} transparent.
\markup {
  \transparent {
    \texttt{invisible text}
  }
}

\verbatim-file \texttt{name (string)}
Read the contents of file \texttt{name}, and include it verbatim.
\markup {
  \verbatim-file #"\texttt{simple.ly}"
}

%%% A simple piece in LilyPond, a scale.
\relative c' {
  c d e f g a b c
}
%%% Optional helper for automatic updating by convert-ly.
%%% May be omitted.
\version "2.16.0"

\whiteout \texttt{arg (markup)}
Provide a white background for \texttt{arg}.
\markup {
  \combine
    \filled-box #'(-1 . 10) #'(-3 . 4) \texttt{\texttt{\whiteout whiteout}}
}

\with-color \texttt{color (color) arg (markup)}
Draw \texttt{arg} in color specified by \texttt{color}. 
A.11 Text markup list commands

The following commands can all be used with \markuptlist:

\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.
\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.12 List of special characters
The following special characters references can be used; for more details, see [ASCII aliases], page 480.

The HTML syntax is used and most of these references are the same as HTML. The rest of them are inspired by \LaTeX.

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.

\&hellip;  \_\_\_  \&ndash;  \_\_\_  \&mdash;  \_\_\_  \&iexcl;  \[]
\&quest;  \[\[\[  \&solidus;  \[\[\[  \&flq;  \[\[\[  \&frq;  \[]
\&flqq;  \[\[\[  \&frqq;  \[\[\[  \&glq;  \[\[\[  \&grq;  \[]
\&glqq;  \[\[\[  \&grqq;  \[\[\[  \&elq;  \[\[\[  \&erq;  \[]
\&elqq;  \[\[\[  \&erqq;  \[\[\[  \&ensp;  \_\_\_  \&emsp;  \_\_\_
Appendix A: Notation manual tables

| &thinsp;   | −  | &nbsp; | −  | &nnbsp; | −  | &zwj; | ． |
| &zwnj;    | ． | &middot; | −  | &bull; |  ©  | &copyright; |  © |
| &registered; |  ©  | &trademark; |  ©  | &dagger; | †  | &Dagger; |  † |
| &numero; |  №  | &ordf; |  ©  | &ordm; |  ©  | &para; |  № |
| &sect; |  §  | &deg; |  ©  | &numero; |  №  | &permil; |  ‡ |
| &brvbar; |  †  | &acute; |  ©  | &acutedbl; |  ©  | &grave; |  © |
| &breve; |  ◌  | &caron; |  ©  | &cedilla; |  ©  | &circumflex; |  ◌ |
| &diaeresis; |  ¯  | &macron; |  ©  | &aa; |  ＠  | &AA; |  ＠ |
| &ae; |  &aEl; | &AE; |  Ｈ  | &dh; |  Ｈ  | &DH; |  Ｄ |
| &dj; |  Ｋ  | &DJ; |  Ｐ  | &l; |  Ｋ  | &L; |  Ｌ |
| &ng; |  Ｎ  | &NG; |  Ｎ  | &o; |  Ｎ  | &O; |  Ｏ |
| &oe; |  Ｎ  | &OE; |  Ｏ  | &s; |  Ｏ  | &ss; |  Ｏ |
| &th; |  Ｔ  | &TH; |  Ｔ  | &plus; |  Ｔ  | &minus; |  － |
| &times; |  ×  | &div; |  ×  | &sup1; |  ×  | &sup2; |  × |
| &sup3; |  ×  | &sqrt; |  √  | &increment; |  √  | &infty; |  √ |
| &sum; |  ∑  | &pm; |  ∑  | &bulletop; |  ∑  | &partial; |  ∑ |
| &neg; |  ＝  | &currency; |  ＝  | &dollar; |  ＝  | &euro; |  ＝ |
| &pounds; |  ¥  | &yen; |  ¥  | &cent; |  ¥  | &cent; |  ¥ |
A.13 List of articulations

The following scripts are available in the Feta font and may be attached to notes (eg. ‘c\accent’).

Articulation scripts

\[
\begin{align*}
\text{accent} & \quad \text{espressivo} & \quad \text{marcato} & \quad \text{portato} \\
\text{staccatissimo} & \quad \text{staccato} & \quad \text{tenuto}
\end{align*}
\]

Ornament scripts

\[
\begin{align*}
\text{prall} & \quad \text{mordent} & \quad \text{prallmordent} & \quad \text{turn} \\
\text{upprall} & \quad \text{downprall} & \quad \text{upmordent} \\
\text{downmordent} & \quad \text{lineprall} & \quad \text{prallprall} \\
\text{pralldown} & \quad \text{prallup} & \quad \text{reverseturn} & \quad \text{trill}
\end{align*}
\]

Fermata scripts

\[
\begin{align*}
\text{shortfermata} & \quad \text{fermata} \\
\text{longfermata} & \quad \text{verylongfermata}
\end{align*}
\]
Instrument-specific scripts

\[ \text{upbow} \quad \text{downbow} \quad \text{flageolet} \quad \text{snappizzicato} \quad \text{open} \]

\[ \text{halfopen} \quad \text{stopped} \quad \text{lheel} \quad \text{rheel} \quad \text{ltoe} \quad \text{rtoe} \]

Repeat sign scripts

\[ \text{segno} \quad \text{coda} \quad \text{varcoda} \]

Ancient scripts

\[ \text{ictus} \quad \text{accentus} \quad \text{circulus} \quad \text{semicirculus} \]

\[ \text{signumcongruentiae} \]

A.14 Percussion notes

- acousticbassdrum: bda
- snare: sn
- acousticsnare: sna

\[ \text{bassdrum: bd} \quad \text{electricsnare: sne} \]

- lowfloortom: toml
- lowtom: toml
- lowmidtom: tomml

\[ \text{highfloortom: tomfh} \quad \text{hightom: tomh} \quad \text{himidtom: tommh} \]

- closedhihat: hhc
- pedalhihat: hhp
- halfopenhihat: hhho

\[ \text{hihat: hh} \quad \text{openhihat: hho} \]

- crashcymbala: cymca
- ridecymbala: cymra

\[ \text{crashcymbal: cymc} \quad \text{ridecymbal: cymr} \]
A.15 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:mi**c**sic?) 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.

A *simple closure* is a closure whose expression has no free variables and hence no free variable bindings.

A simple closure is represented in LilyPond by a smob containing the expression and a method to apply the expression to a passed list of arguments.

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.

See also

Notation Reference: Section 1.8.3 [Fonts], page 236, Section 3.3.3 [Special characters], page 478.

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 O**B**jects’, or *grobs* for short. They are represented by instances of the Grob class.

See also


Internals Reference: Section “grob-interface” in Internals Reference, Section “All layout objects” in Internals Reference.
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.

See also
Notation Reference: [mutable], page 703.

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.

See also

Internals Reference: Section 5.2.2 [Layout interfaces], page 564.

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], page 703. 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.

See also
Notation Reference: [immutable], page 703.

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 Section “LilyPond grammar” in Contributor’s Guide. 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**

PProperty 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.

**simple closure**

See [closure], page 702.

**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.
See also

Internals Reference: Section “grob-interface” in Internals Reference.

A.16 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.

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.

associatedVoice (string)
   Name of the Voice 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 Section “Score” in Internals Reference then all staves share accidentals, and if context is Section “Staff” in Internals Reference 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.

bassStaffProperties (list)
   An alist of property settings to apply for the down staff of PianoStaff. Used by \autochange.

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)
Determine the way the ClefModifier grob is displayed. Possible values are
‘default’, ‘parenthesized’ and ‘bracketed’.

completionBusy (boolean)
Whether a completion-note head is playing.

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)
Determine 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 Section “Timing translator” in Internals Reference at
Section “Score” in Internals Reference 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 ‘:1.S.1:’.

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)
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.

doubleSegnoType (string)
Set the default bar line for the combinations ending of repeat with segno. Default is ‘:1.S’.

endRepeatSegnoType (string)
Set the default bar line for the endings 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) (source1 . 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 Section “Lyrics” in Internals Reference line.
implicitBassFigures (list)
   A list of bass figures that are not printed as numbers, but only as extender lines.

implicitTimeSignatureVisibility (vector)
   break visibility for the default time signature.

includeGraceNotes (boolean)
   Do not ignore grace notes for Section “Lyrics” in Internals Reference.

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 \quote.

internalBarNumber (integer)
   Contains the current bar number. 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).

keySignature (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. keySignature = #`((6 . ,FLAT)).

lyricMelismaAlignment (number)
   Alignment to use for a melisma syllable.

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).

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.

**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.

**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 propor-
tional 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 cor-
responding 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.

{\set Score.skipBars= ##t
 r1 r1*3 R1*3
 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 Section “Pitch_squash_engraver” in Internals Reference.

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.

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

trebleStaffProperties (list)
   An alist of property settings to apply for the up staff of PianoStaff. Used by \autochange.

tremoloFlags (integer)
   The number of tremolo flags to add if no number is specified.

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.17 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 encom-
passes.

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 beam-
let 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 break-align grob will have their X-offsets shifted by this num-
  ber. 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 align-
  ing an anchor to a grob’s extent.

break-align-orders (vector)
  Defines the order in which prefatory matter (clefs, key signatures) appears. The
  format is a vector of length 3, where each element is one order for end-of-line,
  middle of line, and start-of-line, respectively. An order is a list of symbols.
  For example, clefs are put after key signatures by setting

  \override Score.BreakAlignment #'break-align-orders =
  #(make-vector 3 '(span-bar
                   breathing-sign
                   staff-bar
                   key
                   clef
                   time-signature))

break-align-symbol (symbol)
  This key is used for aligning and spacing breakable items.

break-align-symbols (list)
  A list of symbols that determine which break-aligned grobs 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 left-edge,
  ambitus, breathing-sign, clef, staff-bar, key-cancellation, key-signature,
  time-signature, and custos.

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.

circled-tip (boolean)
    Put a circle at start/end of hairpins (al/del niente).

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-bias (number)
    Number determining how much to favor the left (negative) or right (positive). Larger absolute values in either direction will push a collision in this direction.

collision-interfaces (list)
    A list of interfaces for which automatic beam-collision resolution is run.

collision-padding (number)
    Amount of padding to apply after a collision is detected via the self-alignment-interface.

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)
    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 0.0 (no line) and 1.0 (continuous line).

**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)
A list 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.

**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.

**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. 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 Section “note-collision-interface” in Internals Reference.

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.
   • 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".
Appendix A: Notation manual tables

- **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 $\text{thickness} \times (1+\text{string-thickness-factor})^k$. 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?
Appendix A: Notation manual tables

**hair-thickness (number)**
Thickness of the thin line in a bar line.

**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 Section “note-collision-interface” in Internals Reference.

**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-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)
   Amount of extra white space to add. For bar lines, this is the amount of space after a thick line.

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.

labels (list)
   List of labels (symbols) placed on a column.

layer (integer)
   An integer which determines the order of printing objects. Objects with the lowest value of layer are drawn first, then objects with progressively higher values are drawn, so objects with higher values overwrite objects with lower values. By default most objects are assigned a layer value of 1.

ledger-extra (dimension, in staff space)
   Extra distance from staff line to draw ledger lines for.

ledger-line-thickness (pair of numbers)
   The thickness of ledger lines. It is the sum of 2 numbers: The first is the factor for line thickness, and the second for staff space. Both contributions are added.

ledger-positions (list)
   Repeating pattern for the vertical positions of ledger lines. Bracketed groups are always shown together.

left-bound-info (list)
   An alist of properties for determining attachments of spanners to edges.

left-padding (dimension, in staff space)
   The amount of space that is put left to an object (e.g., a lyric extender).

length (dimension, in staff space)
   User override for the stem length of unbeamed stems.

length-fraction (number)
   Multiplier for lengths. Used for determining ledger lines and stem lengths.

line-break-penalty (number)
   Penalty for a line break at this column. This affects the choices of the line breaker; it avoids a line break at a column with a positive penalty and prefers a line break at a column with a negative penalty.
line-break-permission (symbol)
   Instructs the line breaker on whether to put a line break at this column. Can be
   force or allow.

line-break-system-details (list)
   An alist of properties to use if this column is the start of a system.

line-count (integer)
   The number of staff lines.

line-positions (list)
   Vertical positions of staff lines.

line-thickness (number)
   The thickness of the tie or slur contour.

long-text (markup)
   Text markup. See Section “Formatting text” in Notation Reference.

max-beam-connect (integer)
   Maximum number of beams to connect to beams from this stem. Further beams
   are typeset as beamlets.

max-stretch (number)
   The maximum amount that this VerticalAxisGroup can be vertically stretched
   (for example, in order to better fill a page).

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 Section “note-collision-interface” in Internals
   Reference.
   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-fraction (number)
   Minimum length of ledger line as fraction of note head size.
Appendix A: Notation manual tables

minimum-space (dimension, in staff space)
    Minimum distance that the victim should move (after padding).

minimum-X-extent (pair of numbers)
    Minimum size of an object in X dimension, measured in staff-space units.

minimum-Y-extent (pair of numbers)
    Minimum size of an object in Y dimension, measured in staff-space units.

neutral-direction (direction)
    Which direction to take in the center of the staff.

neutral-position (number)
    Position (in half staff spaces) where to flip the direction of custos stem.

next (graphical (layout) object)
    Object that is next relation (e.g., the lyric syllable following an extender).

no-alignment (boolean)
    If set, don’t place this grob in a VerticalAlignment; rather, place it using its own Y-offset callback.

no-ledgers (boolean)
    If set, don’t draw ledger lines on this object.

no-stem-extend (boolean)
    If set, notes with ledger lines do not get stems extending to the middle staff line.

non-break-align-symbols (list)
    A list of symbols that determine which NON-break-aligned interfaces to align this to.

non-default (boolean)
    Set for manually specified clefs.

non-musical (boolean)
    True if the grob belongs to a NonMusicalPaperColumn.

nonstaff-nonstaff-spacing (list)
    The spacing alist controlling the distance between the current non-staff line and the next non-staff line in the direction of staff-affinity, if both are on the same side of the related staff, and staff-affinity is either UP or DOWN. See staff-staff-spacing for a description of the alist structure.

nonstaff-relatedstaff-spacing (list)
    The spacing alist controlling the distance between the current non-staff line and the nearest staff in the direction of staff-affinity, if there are no non-staff lines between the two, and staff-affinity is either UP or DOWN. If staff-affinity is CENTER, then nonstaff-relatedstaff-spacing is used for the nearest staves on both sides, even if other non-staff lines appear between the current one and either of the staves. See staff-staff-spacing for a description of the alist structure.

nonstaff-unrelatedstaff-spacing (list)
    The spacing alist controlling the distance between the current non-staff line and the nearest staff in the opposite direction from staff-affinity, if there are no other non-staff lines between the two, and staff-affinity is either UP or DOWN. See staff-staff-spacing for a description of the alist structure.

normalized-endpoints (pair)
    Represents left and right placement over the total spanner, where the width of the spanner is normalized between 0 and 1.
note-names (vector)
Vector of strings containing names for easy-notation note heads.

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.

parenthesized (boolean)
Parenthesize this grob.
Appendix A: Notation manual tables

**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?

**replacement-alist** (list)
A list 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.

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.

self-alignment-Y (number)
Like self-alignment-X but for the Y axis.

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 (dimension, in staff space)
Start with this much space for the shortest duration. This is expressed in spacing-increment as unit. See also Section “spacing-spanner-interface” in Internals Reference.

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)
Size of object, relative to standard 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)
  A table that specifies distances between prefatory items, like clef and time-signature. The format is an alist of spacing tuples: (break-align-symbol type . distance), where type can be the symbols minimum-space or extra-space.

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 (number)
  Add this much space for a doubled duration. Typically, the width of a note head. See also Section “spacing-spanner-interface” in Internals Reference.

spacing-pair (pair)
  A pair of alignment symbols which set an object’s spacing relative to its left and right BreakAlignments.
  For example, a MultiMeasureRest will ignore prefatory items at its bounds (i.e., clefs, key signatures and time signatures) using the following override:
  \override MultiMeasureRest
  #'spacing-pair = #'(staff-bar . staff-bar)

spanner-id (string)
  An identifier to distinguish concurrent spanners.

springs-and-rods (boolean)
  Dummy variable for triggering spacing routines.

stacking-dir (direction)
  Stack objects in which direction?

staff-affinity (direction)
  The direction of the staff to use for spacing the current non-staff line. Choices are UP, DOWN, and CENTER. If CENTER, the non-staff line will be placed equidistant between the two nearest staves on either side, unless collisions or other spacing constraints prevent this. Setting staff-affinity for a staff causes it to be treated as a non-staff line. Setting staff-affinity to #f causes a non-staff line to be treated as a staff.

staff-padding (dimension, in staff space)
  Maintain this much space between reference points and the staff. Its effect is to align objects of differing sizes (like the dynamics p and f) on their baselines.
**staff-position** (number)
Vertical position, measured in half staff spaces, counted from the middle line.

**staff-space** (dimension, in staff space)
Amount of space between staff lines, expressed in global staff-space.

**staff-staff-spacing** (list)
When applied to a staff-group’s StaffGrouper grob, this spacing alist controls the distance between consecutive staves within the staff-group. When applied to a staff’s VerticalAxisGroup grob, it controls the distance between the staff and the nearest staff below it in the same system, replacing any settings inherited from the StaffGrouper grob of the containing staff-group, if there is one. This property remains in effect even when non-staff lines appear between staves. The alist can contain the following keys:

- **basic-distance** – the vertical distance, measured in staff-spaces, between the reference points of the two items when no collisions would result, and no stretching or compressing is in effect.
- **minimum-distance** – the smallest allowable vertical distance, measured in staff-spaces, between the reference points of the two items, when compressing is in effect.
- **padding** – the minimum required amount of unobstructed vertical whitespace between the bounding boxes (or skylines) of the two items, measured in staff-spaces.
- **stretchability** – a unitless measure of the dimension’s relative propensity to stretch. If zero, the distance will not stretch (unless collisions would result).

**staffgroup-staff-spacing** (list)
The spacing alist controlling the distance between the last staff of the current staff-group and the staff just below it in the same system, even if one or more non-staff lines exist between the two staves. If the staff-staff-spacing property of the staff’s VerticalAxisGroup grob is set, that is used instead. See staff-staff-spacing for a description of the alist structure.

**stem-attachment** (pair of numbers)
An \((x . y)\) pair where the stem attaches to the notehead.

**stem-begin-position** (number)
User override for the begin position of a stem.

**stem-spacing-correction** (number)
Optical correction amount for stems that are placed in tight configurations. For opposite directions, this amount is the correction for two normal sized stems that overlap completely.

**stemlet-length** (number)
How long should be a stem over a rest?

**stencil** (stencil)
The symbol to print.

**stencils** (list)
Multiple stencils, used as intermediate value.

**strict-grace-spacing** (boolean)
If set, main notes are spaced normally, then grace notes are put left of the musical columns for the main notes.
strict-note-spacing (boolean)
    If set, unbroken columns with non-musical material (clefs, bar lines, etc.) are not
    spaced separately, but put before musical columns.

stroke-style (string)
    Set to "grace" to turn stroke through flag on.

style (symbol)
    This setting determines in what style a grob is typeset. Valid choices depend on the
    stencil callback reading this property.

text (markup)
    Text markup. See Section “Formatting text” in Notation Reference.

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)
    Bar line thickness, measured in line-thickness.

thickness (number)
    Line thickness, generally measured in line-thickness.

thin-kern (number)
    The space after a hair-line in a bar line.

tie-configuration (list)
    List of (position . dir) pairs, indicating the desired tie configuration, where pos-
    ition 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 keep the default position (centered on the note
    head), 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 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.

when (moment)
    Global time step associated with this column happen?
whiteout (boolean)
    If true, the grob is printed over a white background to white-out underlying material,
    if the grob is visible. Usually #f by default.

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-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.18 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 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 con-
    text. Otherwise, the context defaults to ‘Staff’, except for piano styles, which use
    ‘GrandStaff’ as a context.

addChordShape [void] - key-symbol (symbol) tuning (pair) shape-definition (string or pair)
    Add chord shape shape-definition to the chord-shape-table hash with the key (cons
    key-symbol tuning).

addInstrumentDefinition [void] - name (string) lst (list)
    Create instrument name with properties list.

addQuote [void] - name (string) music (music)
    Define music as a quotable music expression named name
**afterGrace** [music] - *main* (music) *grace* (music)
Create grace note(s) after a *main* music expression.

**allowPageTurn** [music]
Allow a page turn. May be used at toplevel (ie between scores or markups), or inside a score.

**allowVoltaHook** [void] - *bar* (string)
(undocumented; fixme)

**alterBroken** [music] - *property* (symbol list or symbol) *arg* (list) *item* (symbol list or music)
Override *property* for pieces of broken spanner *item* with values *arg*. *item* may either be music in the form of a starting spanner event, or a symbol list in the form `Context.Grob` or just `Grob`. If *item* is in the form of a spanner event, *property* may also have the form `Grob.property` for specifying a directed tweak.

**appendToTag** [music] - *tag* (symbol) *more* (music) *music* (music)
Append *more* to the elements of all music expressions in *music* that are tagged with *tag*.

**applyContext** [music] - *proc* (procedure)
Modify context properties with Scheme procedure *proc*.

**applyMusic** [music] - *func* (procedure) *music* (music)
Apply procedure *func* to *music*.

**applyOutput** [music] - *ctx* (symbol) *proc* (procedure)
Apply function *proc* to every layout object in context *ctx*.

**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] - *music* (music)
Make voices that switch between staves automatically.

**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*.

**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 enumerator. 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).

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] - music (music)
   Display the LilyPond input representation of music to the console.

displayMusic [music] - music (music)
   Display the internal representation of music to the console.

displayScheme (any type) - expr (any type)
   Display the internal representation of expr to the console.

displayScheme (any type) - expr (any type)
   Display the internal representation of expr to the console.

displayScheme (any type) - expr (any type)
   Display the internal representation of expr to the console.

displayScheme (any type) - expr (any type)
   Display the internal representation of expr 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.

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.

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] - tag (symbol list or symbol) music (music)
Include only elements of music that are either untagged or tagged with one of the tags in tag. tag may be either a single symbol or a list of symbols.

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 list.

languageSaveAndChange [void] - language (string)
Store the previous pitchnames list, and set a new one.

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 [any type]
Make the music for the \mark command.

modalInversion [music] - around (pitch) to (pitch) scale (music) music (music)
Invert music about around using scale and transpose from around to to.

modalTranspose [music] - from (pitch) to (pitch) scale (music) music (music)
Transpose music from pitch from to pitch to using scale.

musicMap [music] - proc (procedure) mus (music)
Apply proc to mus and all of the music it contains.

noPageBreak [music]
Forbid a page break. May be used at toplevel i.e., between scores or markups, or inside a score.

noPageTurn [music]
Forbid a page turn. May be used at toplevel i.e., between scores or markups, or inside a score.

octaveCheck [music] - pitch (pitch)
Octave check.

offset [music] - property (symbol list or symbol) offsets (any type) item (symbol list or music)
Offset the default value of property of item by offsets. If item is a string, the result is \override for the specified grob type. If item is a music expression, the result is the same music expression with an appropriate tweak applied.
omit [music] - item (symbol list or music)
Set item’s ‘stencil’ property to #f, effectively omitting it without taking up space.
If item is a symbol list of form GrobName or Context.GrobName, the result is an
override for the grob name specified by it. If item is a music expression, the result
is the same music expression with an appropriate tweak applied to it.

once [music] - music (music)
Set once to #t on all layout instruction events in music. This will complain about
music with an actual duration. As a special exception, if music contains ‘tweaks’ it
will be silently ignored in order to allow for \once \tweak to work as both one-time
override and proper tweak.

ottava [music] - octave (integer)
Set the octavation.

overrideProperty [music] - grob-property-path (symbol list) value (any type)
Set the grob property specified by grob-property-path to value. grob-property-path
is a symbol list of the form Context.GrobName.property or GrobName.property,
possibly with subproperties given as well.

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 | }
parenthesize [music] - arg (music)
Tag arg to be parenthesized.

partcombine [music] - part1 (music) part2 (music)
Take the music in part1 and part2 and typeset so that they share a staff.
partcombineDown \[\text{music}\] - \textit{part1} (\text{music}) \textit{part2} (\text{music})

Take the music in \textit{part1} and \textit{part2} and typeset so that they share a staff with stems directed downward.

partcombineForce \[\text{music}\] - type (\text{symbol-or-boolean}) \textit{once} (\text{boolean})

Override the part-combiner.

partcombineUp \[\text{music}\] - \textit{part1} (\text{music}) \textit{part2} (\text{music})

Take the music in \textit{part1} and \textit{part2} and typeset so that they share a staff with stems directed upward.

partial \[\text{music}\] - \textit{dur} (\text{duration})

Make a partial measure.

phrasingSlurDashPattern \[\text{music}\] - \textit{dash-fraction} (\text{number}) \textit{dash-period} (\text{number})

Set up a custom style of dash pattern for \textit{dash-fraction} ratio of line to space repeated at \textit{dash-period} interval for phrasing slurs.

pitchedTrill \[\text{music}\] - \textit{main-note} (\text{music}) \textit{secondary-note} (\text{music})

Print a trill with \textit{main-note} as the main note of the trill and print \textit{secondary-note} as a stemless note head in parentheses.

pointAndClickOff \[\text{void}\]

Suppress generating extra code in final-format (e.g. pdf) files to point back to the lilypond source statement.

pointAndClickOn \[\text{void}\]

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 \[\text{void}\] - \textit{types} (\text{symbol list or symbol})

Set a type or list of types (such as \texttt{\#'note-event}) for which point-and-click info is generated.

pushToTag \[\text{music}\] - \textit{tag} (\text{symbol}) \textit{more} (\text{music}) \textit{music} (\text{music})

Add \textit{more} to the front of \textit{elements} of all music expressions in \textit{music} that are tagged with \textit{tag}.

quoteDuring \[\text{music}\] - \textit{what} (\text{string}) \textit{main-music} (\text{music})

Indicate a section of music to be quoted. \textit{what} indicates the name of the quoted voice, as specified in an \texttt{\addQuote} command. \textit{main-music} is used to indicate the length of music to be quoted; usually contains spacers or multi-measure rests.

relative \[\text{music}\] - \textit{pitch} [\textit{pitch}] \textit{music} (\text{music})

Make \textit{music} relative to \textit{pitch}. If \textit{pitch} is omitted, the first note in \textit{music} is given in absolute pitch.

removeWithTag \[\text{music}\] - \textit{tag} (\text{symbol list or symbol}) \textit{music} (\text{music})

Remove elements of \textit{music} that are tagged with one of the tags in \textit{tag}. \textit{tag} may be either a single symbol or a list of symbols.

resetRelativeOctave \[\text{music}\] - \textit{pitch} (\text{pitch})

Set the octave inside a \texttt{\relative} section.

retrograde \[\text{music}\] - \textit{music} (\text{music})

Return \textit{music} in reverse order.

revertTimeSignatureSettings \[\text{music}\] - \textit{time-signature} (\text{pair})

Revert \textit{timeSignatureSettings} for time signatures of \textit{time-signature}.
rightHandFinger [post event] - finger (number or markup)
Apply finger as a fingering indication.

scaleDurations [music] - fraction (fraction, as pair) music (music)
Multiply the duration of events in music by fraction.

settingsFrom (any type) - ctx [symbol] music (music)
Take the layout instruction events from music, optionally restricted to those applying to context type ctx, and return a context modification duplicating their effect.

shape [music] - offsets (list) item (symbol list or music)
Offset control-points of item by offsets. The argument is a list of number pairs or list of such lists. Each element of a pair represents an offset to one of the coordinates of a control-point. If item is a string, the result is \once\override for the specified grob type. If item is a music expression, the result is the same music expression with an appropriate tweak applied.

shiftDurations [music] - dur (integer) dots (integer) arg (music)
Change the duration of arg by adding dur to the durlog of arg and dots to the dots of arg.

single [music] - overrides (music) music (music)
Convert overrides to tweaks and apply them to music. This does not convert \revert, \set or \unset.

skip [music] - dur (duration)
Skip forward by dur.

slashedGrace [music] - music (music)
Create slashed graces (slashes through stems, but no slur) from the following music expression

slurDashPattern [music] - dash-fraction (number) dash-period (number)
Set up a custom style of dash pattern for dash-fraction ratio of line to space repeated at dash-period interval for slurs.

spacingTweaks [music] - parameters (list)
Set the system stretch, by reading the 'system-stretch property of the 'parameters' assoc list.

storePredefinedDiagram [void] - fretboard-table (hash table) chord (music) tuning (pair)
diagram-definition (string or pair)
Add predefined fret diagram defined by diagram-definition for the chord pitches chord and the stringTuning tuning.

stringTuning (any type) - chord (music)
Convert chord to a string tuning. chord must be in absolute pitches and should have the highest string number (generally the lowest pitch) first.

styledNoteHeads [music] - style (symbol) heads (symbol list or symbol) music (music)
Set heads in music to style.

tabChordRepeats [music] - event-types [list] music (music)
Walk through music putting the notes, fingerings and string numbers of the previous chord into repeat chords, as well as an optional list of event-types such as #'(articulation-event).

tabChordRepetition [void]
Include the string and fingering information in a chord repetition. This function is deprecated; try using \tabChordRepeats instead.
tag [music] - tag (symbol list or symbol) music (music)
Tag the following music with tag and return the result, by adding the single symbol or symbol list tag to the tags property of music.

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.

transposeCueDuring [music] - what (string) dir (direction) pitch (pitch) main-music (music)
Insert notes from the part what into a voice called cue, using the transposition defined by pitch. This happens simultaneously with main-music, which is usually a rest. The argument dir determines whether the cue notes should be notated as a first or second voice.

transposition [music] - pitch (pitch)
Set instrument transposition

tuplet [music] - ratio (fraction, as pair) tuplet-span [duration] music (music)
Scale the given music to tuplets. ratio is a fraction that specifies how many notes are played in place of the nominal value: it will be ‘3/2’ for triplets, namely three notes being played in place of two. If the optional duration tuplet-span is specified, it is used instead of tupletSpannerDuration for grouping the tuplets. For example, \tuplet 3/2 4 { c8 c c c c c }
will result in two groups of three tuplets, each group lasting for a quarter note.

tupletSpan [music] - tuplet-span [duration]
Set tupletSpannerDuration, the length into which \tuplet without an explicit ‘tuplet-span’ argument of its own will group its tuplets, to the duration tuplet-span. To revert to the default of not subdividing the contents of a \tuplet command without explicit ‘tuplet-span’, use \tupletSpan \default

tweak [music] - prop (symbol list or symbol) value (any type) item (symbol list or music)
Add a tweak to the following item, usually music. Layout objects created by item get their property prop set to value. If prop has the form ‘Grob.property’, like with
an indirectly created grob (‘Accidental’ is caused by ‘NoteHead’) can be tweaked; otherwise only directly created grobs are affected.

As a special case, item may be a symbol list specifying a grob path, in which case \override is called on it instead of creating tweaked music. This is mainly useful when using \tweak as as a component for building other functions.

If this use case would call for \once \override rather than a plain \override, writing \once \tweak ... can be convenient.

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.

A.19 Context modification identifiers

The following commands are defined for use as context modifications within a \layout or \with block.

RemoveEmptyStaves
Remove staves which are considered to be empty according to the list of interfaces set by keepAliveInterfaces.

- Sets grob property remove-empty in Section ‘‘VerticalAxisGroup’’ in Internals Reference to #t.

A.20 Predefined type predicates

R5RS primary predicates

<table>
<thead>
<tr>
<th>Type predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean?</td>
<td>boolean</td>
</tr>
<tr>
<td>char?</td>
<td>character</td>
</tr>
<tr>
<td>number?</td>
<td>number</td>
</tr>
<tr>
<td>pair?</td>
<td>pair</td>
</tr>
<tr>
<td>port?</td>
<td>port</td>
</tr>
<tr>
<td>procedure?</td>
<td>procedure</td>
</tr>
<tr>
<td>String</td>
<td>Symbol</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
</tr>
</tbody>
</table>

### R5RS secondary predicates

<table>
<thead>
<tr>
<th>Type predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char-alphabetic?</td>
<td>alphabetic character</td>
</tr>
<tr>
<td>char-lower-case?</td>
<td>lower-case character</td>
</tr>
<tr>
<td>char-numeric?</td>
<td>numeric character</td>
</tr>
<tr>
<td>char-upper-case?</td>
<td>upper-case character</td>
</tr>
<tr>
<td>char-whitespace?</td>
<td>whitespace character</td>
</tr>
<tr>
<td>complex?</td>
<td>complex number</td>
</tr>
<tr>
<td>eof-object?</td>
<td>end-of-file object</td>
</tr>
<tr>
<td>even?</td>
<td>even number</td>
</tr>
<tr>
<td>exact?</td>
<td>exact number</td>
</tr>
<tr>
<td>inexact?</td>
<td>inexact number</td>
</tr>
<tr>
<td>input-port?</td>
<td>input port</td>
</tr>
<tr>
<td>integer?</td>
<td>integer</td>
</tr>
<tr>
<td>list?</td>
<td>list (use cheap-list? for faster processing)</td>
</tr>
<tr>
<td>negative?</td>
<td>negative number</td>
</tr>
<tr>
<td>null?</td>
<td>null</td>
</tr>
<tr>
<td>odd?</td>
<td>odd number</td>
</tr>
<tr>
<td>output-port?</td>
<td>output port</td>
</tr>
<tr>
<td>positive?</td>
<td>positive number</td>
</tr>
<tr>
<td>rational?</td>
<td>rational number</td>
</tr>
<tr>
<td>real?</td>
<td>real number</td>
</tr>
<tr>
<td>zero?</td>
<td>zero</td>
</tr>
</tbody>
</table>

### Guile predicates

<table>
<thead>
<tr>
<th>Type predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hash-table?</td>
<td>hash table</td>
</tr>
</tbody>
</table>

### LilyPond scheme predicates

<table>
<thead>
<tr>
<th>Type predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean-or-symbol?</td>
<td>boolean or symbol</td>
</tr>
<tr>
<td>cheap-list?</td>
<td>list (use this instead of list? for faster processing)</td>
</tr>
<tr>
<td>color?</td>
<td>color</td>
</tr>
<tr>
<td>fraction?</td>
<td>fraction, as pair</td>
</tr>
<tr>
<td>grob-list?</td>
<td>list of grobs</td>
</tr>
<tr>
<td>index?</td>
<td>non-negative integer</td>
</tr>
<tr>
<td>markup?</td>
<td>markup</td>
</tr>
<tr>
<td>markup-command-list?</td>
<td>markup command list</td>
</tr>
<tr>
<td>markup-list?</td>
<td>markup list</td>
</tr>
<tr>
<td>moment-pair?</td>
<td>pair of moment objects</td>
</tr>
<tr>
<td>number-list?</td>
<td>number list</td>
</tr>
<tr>
<td>number-or-grob?</td>
<td>number or grob</td>
</tr>
<tr>
<td>number-or-markup?</td>
<td>number or markup</td>
</tr>
<tr>
<td>number-or-pair?</td>
<td>number or pair</td>
</tr>
</tbody>
</table>
number-or-string?  number or string
number-pair?  pair of numbers
number-pair-list?  list of number pairs
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

<table>
<thead>
<tr>
<th>Type predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ly:box?</td>
<td>box</td>
</tr>
<tr>
<td>ly:context?</td>
<td>context</td>
</tr>
<tr>
<td>ly:context-def?</td>
<td>context definition</td>
</tr>
<tr>
<td>ly:context-mod?</td>
<td>context modification</td>
</tr>
<tr>
<td>ly:dimension?</td>
<td>dimension, in staff space</td>
</tr>
<tr>
<td>ly:dir?</td>
<td>direction</td>
</tr>
<tr>
<td>ly:dispatcher?</td>
<td>dispatcher</td>
</tr>
<tr>
<td>ly:duration?</td>
<td>duration</td>
</tr>
<tr>
<td>ly:event?</td>
<td>post event</td>
</tr>
<tr>
<td>ly:font-metric?</td>
<td>font metric</td>
</tr>
<tr>
<td>ly:grob?</td>
<td>graphical (layout) object</td>
</tr>
<tr>
<td>ly:grob-array?</td>
<td>array of grobs</td>
</tr>
<tr>
<td>ly:input-location?</td>
<td>input location</td>
</tr>
<tr>
<td>ly:item?</td>
<td>item</td>
</tr>
<tr>
<td>ly:iterator?</td>
<td>iterator</td>
</tr>
<tr>
<td>ly:lily-lexer?</td>
<td>lily-lexer</td>
</tr>
<tr>
<td>ly:lily-parser?</td>
<td>lily-parser</td>
</tr>
<tr>
<td>ly:listener?</td>
<td>listener</td>
</tr>
<tr>
<td>ly:moment?</td>
<td>moment</td>
</tr>
<tr>
<td>ly:music?</td>
<td>music</td>
</tr>
<tr>
<td>ly:music-function?</td>
<td>music function</td>
</tr>
<tr>
<td>ly:music-list?</td>
<td>list of music objects</td>
</tr>
<tr>
<td>ly:music-output?</td>
<td>music output</td>
</tr>
<tr>
<td>ly:otf-font?</td>
<td>OpenType font</td>
</tr>
<tr>
<td>ly:output-def?</td>
<td>output definition</td>
</tr>
<tr>
<td>ly:page-marker?</td>
<td>page marker</td>
</tr>
<tr>
<td>ly:pango-font?</td>
<td>pango font</td>
</tr>
<tr>
<td>ly:paper-system?</td>
<td>paper-system Prob</td>
</tr>
<tr>
<td>ly:pitch?</td>
<td>pitch</td>
</tr>
<tr>
<td>ly:prob?</td>
<td>property object</td>
</tr>
<tr>
<td>ly:score?</td>
<td>score</td>
</tr>
<tr>
<td>ly:simple-closure?</td>
<td>simple closure</td>
</tr>
<tr>
<td>ly:skyline?</td>
<td>skyline</td>
</tr>
</tbody>
</table>
Appendix A: Notation manual tables

ly:skyline-pair?  pair of skylines
ly:source-file?  source file
ly:spanner?  spanner
ly:spring?  spring
ly:stencil?  stencil
ly:stream-event?  stream event
ly:translator?  translator
ly:translator-group?  translator group
ly:unpure-pure-container?  unpure/pure container

A.21 Scheme functions

ly:add-context-mod contextmods modification  [Function]
Adds the given context modification to the list contextmods of context modifications.

ly:add-file-name-alist alist  [Function]
Add mappings for error messages from alist.

ly:add-interface iface desc props  [Function]
Add a new grob interface. iface is the interface name, desc is the interface description, and props is the list of user-settable properties for the interface.

ly:add-listener list disp cl  [Function]
Add the listener list to the dispatcher disp. Whenever disp hears an event of class cl, it is forwarded to list.

ly:add-option sym val description  [Function]
Add a program option sym. val is the default value and description is a string description.

ly:all-grob-interfaces  [Function]
Return the hash table with all grob interface descriptions.

ly:all-options  [Function]
Get all option settings in an alist.

ly:all-stencil-expressions  [Function]
Return all symbols recognized as stencil expressions.

ly:assoc-get key alist default-value strict-checking  [Function]
Return value if key in alist, else default-value (or #f if not specified). If strict-checking is set to #t and key is not in alist, a programming_error is output.

ly:axis-group-interface::add-element grob grob-element  [Function]
Set grob the parent of grob-element on all axes of grob.

ly:basic-progress str rest  [Function]
A Scheme callable function to issue a basic progress message str. The message is formatted with format and rest.

ly:beam-score-count  [Function]
count number of beam scores.

ly:book? x  [Function]
Is x a Book object?

Appendix A: Notation manual tables

ly:book-add-score!  book-smob score
  Add score to book-smob score list.


  Return header in book.

  Return paper in book.

  Print book. output is passed to the backend unchanged. For example, it may be a string (for file based outputs) or a socket (for network based output).

  Print book. output 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  book
  Return scores in book.

ly:box?  x
  Is x a Box object?

ly:bp  num
  num bigpoints (1/72th inch).

ly:bracket  a iv t p
  Make a bracket in direction a. The extent of the bracket is given by iv. The wings protrude by an amount of p, which may be negative. The thickness is given by t.

ly:broadcast  disp ev
  Send the stream event ev to the dispatcher disp.

ly:camel-case->lisp-identifier  name-sym
  Convert FooBar_Bla to foo-bar-bla style symbol.

ly:chain-assoc-get  key achain default-value strict-checking
  Return value for key from a list of alists achain. If no entry is found, return default-value or #f if default-value is not specified. With strict-checking set to #t, a programming_error is output in such cases.

ly:check-expected-warnings
  Check whether all expected warnings have really been triggered.

ly:cm  num
  num cm.

ly:command-line-code
  The Scheme code specified on command-line with ‘-e’.
ly:command-line-options
The Scheme options specified on command-line with \'d\'.

ly:connect-dispatchers to from
Make the dispatcher to listen to events from from.

ly:context? x
Is x a Context object?

ly:context-current-moment context
Return the current moment of context.

ly:context-def? x
Is x a Context_def object?

ly:context-def-lookup def sym val
Return the value of sym in context definition def (e.g., \Voice). If no value is found, return val or \'() if val is undefined. sym can be any of 'default-child', 'consists', 'description', 'aliases', 'accepts', 'property-ops', 'context-name', 'group-type'.

ly:context-def-modify def mod
Return the result of applying the context-mod mod to the context definition def. Does not change def.

ly:context-event-source context
Return event-source of context context.

ly:context-events-below context
Return a stream-distributor that distributes all events from context and all its subcontexts.

ly:context-find context name
Find a parent of context that has name or alias name. Return \#f if not found.

ly:context-grob-definition context name
Return the definition of name (a symbol) within context as an alist.

ly:context-id context
Return the ID string of context, i.e., for \context Voice = "one" ... return the string one.

ly:context-mod? x
Is x a Context_mod object?

ly:context-mod-apply! context mod
Apply the context modification mod to context.

ly:context-name context
Return the name of context, i.e., for \context Voice = "one" ... return the symbol Voice.

ly:context-now context
Return now-moment of context context.

ly:context-parent context
Return the parent of context, \#f if none.

ly:context-property context sym def
Return the value for property sym in context. If def is given, and property value is \'(), return def.
### Appendix A: Notation manual tables

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ly:context-property-where-defined</code> <code>context name</code></td>
<td>Return the context above <code>context</code> where <code>name</code> is defined.</td>
</tr>
<tr>
<td><code>ly:context-pushpop-property</code> <code>context grob eltprop val</code></td>
<td>Do a single <code>override</code> or <code>revert</code> operation in <code>context</code>. The grob definition <code>grob</code> is extended with <code>eltprop</code> (if <code>val</code> is specified) or reverted (if unspecified).</td>
</tr>
<tr>
<td><code>ly:context-set-property!</code> <code>context name val</code></td>
<td>Set value of property <code>name</code> in context <code>context</code> to <code>val</code>.</td>
</tr>
<tr>
<td><code>ly:context-unset-property</code> <code>context name</code></td>
<td>Unset value of property <code>name</code> in context <code>context</code>.</td>
</tr>
<tr>
<td><code>ly:debug str rest</code></td>
<td>A Scheme callable function to issue a debug message <code>str</code>. The message is formatted with <code>format</code> and <code>rest</code>.</td>
</tr>
<tr>
<td><code>ly:default-scale</code></td>
<td>Get the global default scale.</td>
</tr>
<tr>
<td><code>ly:dimension? d</code></td>
<td>Return <code>d</code> as a number. Used to distinguish length variables from normal numbers.</td>
</tr>
<tr>
<td><code>ly:dir? s</code></td>
<td>Is <code>s</code> 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.</td>
</tr>
<tr>
<td><code>ly:dispatcher? x</code></td>
<td>Is <code>x</code> a <code>Dispatcher</code> object?</td>
</tr>
<tr>
<td><code>ly:duration? x</code></td>
<td>Is <code>x</code> a <code>Duration</code> object?</td>
</tr>
<tr>
<td><code>ly:duration&lt;? p1 p2</code></td>
<td>Is <code>p1</code> shorter than <code>p2</code>?</td>
</tr>
<tr>
<td><code>ly:duration-&gt;string</code> <code>dur</code></td>
<td>Convert <code>dur</code> to a string.</td>
</tr>
<tr>
<td><code>ly:duration-dot-count</code> <code>dur</code></td>
<td>Extract the dot count from <code>dur</code>.</td>
</tr>
<tr>
<td><code>ly:duration-factor</code> <code>dur</code></td>
<td>Extract the compression factor from <code>dur</code>. Return it as a pair.</td>
</tr>
<tr>
<td><code>ly:duration-length</code> <code>dur</code></td>
<td>The length of the duration as a moment.</td>
</tr>
<tr>
<td><code>ly:duration-log</code> <code>dur</code></td>
<td>Extract the duration log from <code>dur</code>.</td>
</tr>
<tr>
<td><code>ly:duration-scale</code> <code>dur</code></td>
<td>Extract the compression factor from <code>dur</code>. Return it as a rational.</td>
</tr>
<tr>
<td><code>ly:effective-prefix</code></td>
<td>Return effective prefix.</td>
</tr>
</tbody>
</table>
ly:encode-string-for-pdf str
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
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
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
A Scheme callable function to issue the error str. The error is formatted with format and rest.

ly:eval-simple-closure delayed closure scm-start scm-end
Evaluate a simple closure with the given delayed argument. If scm-start and scm-end are defined, evaluate it purely with those start and end points.

ly:event? obj
Is obj a proper (non-rhythmic) event object?

ly:event-deep-copy m
Copy m and all sub expressions of m.

ly:event-property sev sym val
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
Set property sym in event ev to val.

ly:expand-environment str
Expand $VAR and ${VAR} in str.

ly:expect-warning str rest
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
Return the absolute file name of name, or #f if not found.

ly:font-config-add-directory dir
Add directory dir to FontConfig.

ly:font-config-add-font font
Add font font to FontConfig.

ly:font-config-display-fonts
Dump a list of all fonts visible to FontConfig.

ly:font-config-get-font-file name
Get the file for font name.
Appendix A: Notation manual tables

ly:font-design-size font [Function]
Given the font metric font, return the design size, relative to the current output-scale.

ly:font-file-name font [Function]
Given the font metric font, return the corresponding file name.

ly:font-get-glyph font name [Function]
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 [Function]
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 [Function]
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 [Function]
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 [Function]
Given the font metric font, return the magnification, relative to the current output-scale.

ly:font-metric? x [Function]
Is x a Font_metric object?

ly:font-name font [Function]
Given the font metric font, return the corresponding name.

ly:font-sub-fonts font [Function]
Given the font metric font of an OpenType font, return the names of the subfonts within font.

ly:format str rest [Function]
LilyPond specific format, supporting ~a and ~[0-9]f. Basic support for ~s is also provided.

ly:format-output context [Function]
Given a global context in its final state, process it and return the Music_output object in its final state.

ly:get-all-function-documentation [Function]
Get a hash table with all LilyPond Scheme extension functions.

ly:get-all-translators [Function]
Return a list of all translator objects that may be instantiated.
ly:get-context-mods contextmod  [Function]
Returns the list of context modifications stored in contextmod.

ly:get-option var  [Function]
Get a global option setting.

ly:get-spacing-spec from-scm to-scm  [Function]
Return the spacing spec going between the two given grobs, from_scm and to_scm.

ly:get-undead undead  [Function]
Get back object from undead.

ly:gettext original  [Function]
A Scheme wrapper function for gettext.

ly:grob? x  [Function]
Is x a Grob object?

ly:grob-alist-chain grob global  [Function]
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  [Function]
Is x a Grob_array object?

ly:grob-array->list grob-arr  [Function]
Return the elements of grob-arr as a Scheme list.

ly:grob-array-length grob-arr  [Function]
Return the length of grob-arr.

ly:grob-array-ref grob-arr index  [Function]
Retrieve the indexth element of grob-arr.

ly:grob-basic-properties grob  [Function]
Get the immutable properties of grob.

ly:grob-chain-callback grob proc sym  [Function]
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  [Function]
Find the common refpoint of grob and other for axis.

ly:grob-common-refpoint-of-array grob others axis  [Function]
Find the common refpoint of grob and others (a grob-array) for axis.

ly:grob-default-font grob  [Function]
Return the default font for grob grob.

ly:grob-extent grob refp axis  [Function]
Get the extent in axis direction of grob relative to the grob refp.

ly:grob-get-vertical-axis-group-index grob  [Function]
Get the index of the vertical axis group the grob grob belongs to; return -1 if none is found.

ly:grob-interfaces grob  [Function]
Return the interfaces list of grob grob.
Appendix A: Notation manual tables

ly:grob-layout grob
Get \texttt{layout} definition from grob grob.

ly:grob-object grob sym
Return the value of a pointer in grob grob of property sym. It returns `()` (end-of-list) if sym is undefined in grob.

ly:grob-original grob
Return the unbroken original grob of grob.

ly:grob-parent grob axis
Get the parent of grob. axis is 0 for the X-axis, 1 for the Y-axis.

ly:grob-pq<? a b
Compare two grob priority queue entries. This is an internal function.

ly:grob-properties grob
Get the mutable properties of grob.

ly:grob-property grob sym val
Return the value for property sym of grob. If no value is found, return val or `()` if val is not specified.

ly:grob-property-data grob sym
Return the value for property sym of grob, but do not process callbacks.

ly:grob-pure-height grob refp beg end val
Return the pure height of grob given refpoint refp. If no value is found, return val or `()` if val is not specified.

ly:grob-pure-property grob sym beg end val
Return the pure value for property sym of grob. If no value is found, return val or `()` if val is not specified.

ly:grob-relative-coordinate grob refp axis
Get the coordinate in axis direction of grob relative to the grob refp.

ly:grob-robust-relative-extent grob refp axis
Get the extent in axis direction of grob relative to the grob refp, or (0,0) if empty.

ly:grob-script-priority-less a b
Compare two grobs by script priority. For internal use.

ly:grob-set-nested-property! grob symlist val
Set nested property symlist in grob grob to value val.

ly:grob-set-object! grob sym val
Set sym in grob grob to value val.

ly:grob-set-parent! grob axis parent-grob
Set parent-grob the parent of grob grob in axis axis.

ly:grob-set-property! grob sym val
Set sym in grob grob to value val.

ly:grob-staff-position sg
Return the Y-position of sg relative to the staff.
ly:grob-suicide! grob
   Kill grob.

ly:grob-system grob
   Return the system grob of grob.

ly:grob-translate-axis! grob d a
   Translate grob on axis a over distance d.

ly:grob-vertical? a b
   Does a lie above b on the page?

ly:gulp-file name size
   Read size characters from the file name, and return its contents in a string. If size is undefined, the entire file is read. The file is looked up using the search path.

ly:hash-table-keys tab
   Return a list of keys in tab.

ly:inch num
   num inches.

ly:input-both-locations sip
   Return input location in sip as (file-name first-line first-column last-line last-column).

ly:input-file-line-char-column sip
   Return input location in sip as (file-name line char column).

ly:input-location? x
   Is x an input-location?

ly:input-message sip msg rest
   Print msg as a GNU compliant error message, pointing to the location in sip. msg is interpreted similar to format’s argument, using rest.

ly:input-warning sip msg rest
   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
   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
   Parse expr, feed bits to func with first arg arg1 having offset offset.

ly:intlog2 d
   The 2-logarithm of 1/d.

ly:item? g
   Is g an Item object?

ly:item-break-dir it
   The break status direction of item it. -1 means end of line, 0 unbroken, and 1 beginning of line.
ly:iterator? x
   Is x a Music_iterator object?

ly:lexer-keywords lexer
   Return a list of (KEY . CODE) pairs, signifying the LilyPond reserved words list.

ly:lily-lexer? x
   Is x a Lily_lexer object?

ly:lily-parser? x
   Is x a Lily_parser object?

ly:listened-event-class? disp cl
   Does disp listen to any event type in the list cl?

ly:listened-event-types disp
   Return a list of all event types that disp listens to.

ly:listener? x
   Is x a Listener object?

ly:make-book paper header scores
   Make a \book of paper and header (which may be #f as well) containing \scores.

ly:make-book-part scores
   Make a \bookpart containing \scores.

ly:make-context-mod mod-list
   Creates a context modification, optionally initialized via the list of modifications mod-list.

ly:make-dispatcher
   Return a newly created dispatcher.

ly:make-duration length dotcount num den
   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
   Set up a global interpretation context, using the output block output-def. The context is returned.

ly:make-global-translator global
   Create a translator group and connect it to the global context global. The translator group is returned.

ly:make-listener callback
   Create a listener. Any time the listener hears an object, it will call callback with that object. callback should take exactly one argument.
ly:make-moment \( m g g n g d \)  
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 \( \text{props} \)  
Make a C++ Music object and initialize it with \( \text{props} \).

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 \( \text{signature func} \)  
Make a function to process music, to be used for the parser. \( \text{func} \) is the function, and \( \text{signature} \) describes its arguments. \( \text{signature} \)'s cdr is a list containing either \( \text{ly:music?} \) predicates or other type predicates. Its car is the syntax function to call.

ly:make-music-relative! \( \text{music pitch} \)  
Make \( \text{music} \) relative to \( \text{pitch} \), return final pitch.

ly:make-output-def  
Make an output definition.

ly:make-page-label-marker \( \text{label} \)  
Return page marker with label \( \text{label} \).

ly:make-page-permission-marker \( \text{symbol permission} \)  
Return page marker with page breaking and turning permissions.

ly:make-pango-description-string \( \text{chain size} \)  
Make a PangoFontDescription string for the property alist \( \text{chain} \) at size \( \text{size} \).

ly:make-paper-outputer \( \text{port format} \)  
Create an outpputer that evaluates within output-format, writing to \( \text{port} \).

ly:make-pitch \( \text{octave note alter} \)  
octave is specified by an integer, zero for the octave containing middle C. \( \text{note} \) is a number indexing the global default scale, with 0 corresponding to pitch C and 6 usually corresponding to pitch B. Optional \( \text{alter} \) is a rational number of 200-cent whole tones for alteration.

ly:make-prob \( \text{type init rest} \)  
Create a Prob object.

ly:make-scale \( \text{steps} \)  
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 \( \text{music} \)  
Return score with \( \text{music} \) encapsulated in it.

ly:make-simple-closure \( \text{expr} \)  
Make a simple closure. \( \text{expr} \) should be form of \( (\text{func } a1 a2 \ldots) \), and will be invoked as \( (\text{func delayed-arg } a1 a2 \ldots) \).
ly:make-spring \textit{ideal} \textit{min-dist} \hfill [Function]
Make a spring. \textit{ideal} is the ideal distance of the spring, and \textit{min-dist} is the minimum distance.

ly:make-stencil \textit{expr} \textit{xext} \textit{yext} \hfill [Function]
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 \texttt{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 \texttt{empty-interval} as its value), it is taken to be empty.

ly:make-stream-event \textit{cl} \textit{prop-list} \hfill [Function]
Create a stream event of class \textit{cl} with the given mutable property list.

ly:make-undead \textit{object} \hfill [Function]
This packages \textit{object} in a manner that keeps it from triggering "Parsed object should be dead" messages.

ly:make-unpure-pure-container \textit{unpure} \textit{pure} \hfill [Function]
Make an unpure-pure container. \textit{unpure} should be an unpure expression, and \textit{pure} should be a pure expression. If \textit{pure} is omitted, the value of \textit{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 \textit{str} \textit{rest} \hfill [Function]
A Scheme callable function to issue the message \textit{str}. The message is formatted with \texttt{format} and \textit{rest}.

ly:minimal-breaking \textit{pb} \hfill [Function]
Break (pages and lines) the \texttt{Paper_book} object \textit{pb} without looking for optimal spacing: stack as many lines on a page before moving to the next one.

ly:mm \textit{num} \hfill [Function]
\textit{num} mm.

ly:module->alist \textit{mod} \hfill [Function]
Dump the contents of module \textit{mod} as an alist.

ly:module-copy \textit{dest} \textit{src} \hfill [Function]
Copy all bindings from module \textit{src} into \textit{dest}.

ly:modules-lookup \textit{modules} \textit{sym} \textit{def} \hfill [Function]
Look up \textit{sym} in the list \textit{modules}, returning the first occurrence. If not found, return \textit{def} or \#f if \textit{def} isn’t specified.

ly:moment? \textit{x} \hfill [Function]
Is \textit{x} a \texttt{Moment} object?

ly:moment<? \textit{a} \textit{b} \hfill [Function]
Compare two moments.

ly:moment-add \textit{a} \textit{b} \hfill [Function]
Add two moments.

ly:moment-div \textit{a} \textit{b} \hfill [Function]
Divide two moments.

ly:moment-grace \textit{mom} \hfill [Function]
Extract grace timing as a rational number from \textit{mom}.
ly:moment-grace-denominator \(\text{mom}\)  
Extract denominator from grace timing.

ly:moment-grace-numerator \(\text{mom}\)  
Extract numerator from grace timing.

ly:moment-main \(\text{mom}\)  
Extract main timing as a rational number from \(\text{mom}\).

ly:moment-main-denominator \(\text{mom}\)  
Extract denominator from main timing.

ly:moment-main-numerator \(\text{mom}\)  
Extract numerator from main timing.

ly:moment-mod \(a b\)  
Modulo of two moments.

ly:moment-mul \(a b\)  
Multiply two moments.

ly:moment-sub \(a b\)  
Subtract two moments.

ly:music? \(\text{obj}\)  
Is \(\text{obj}\) a music object?

ly:music-compress \(m \text{factor}\)  
Compress music object \(m\) by moment \(\text{factor}\).

ly:music-deep-copy \(m\)  
Copy \(m\) and all sub expressions of \(m\). \(m\) may be an arbitrary type; cons cells and music are copied recursively.

ly:music-duration-compress \(m \text{factor}\)  
Compress \(m\) by factor \(\text{factor}\), which is a Moment.

ly:music-duration-length \(m\)  
Extract the duration field from \(m\) and return the length.

ly:music-function? \(x\)  
Is \(x\) a music-function?

ly:music-function-extract \(x\)  
Return the Scheme function inside \(x\).

ly:music-function-signature \(x\)  
Return the function signature inside \(x\).

ly:music-length \(m\)  
Get the length of music expression \(m\) and return it as a Moment object.

ly:music-list? \(\text{lst}\)  
Is \(\text{lst}\) a list of music objects?

ly:music-mutable-properties \(m\)  
Return an alist containing the mutable properties of \(m\). The immutable properties are not available, since they are constant and initialized by the make-music function.
Appendix A: Notation manual tables 760

ly:music-output? x
   Is x a Music_output object?  [Function]

ly:music-property mus sym val
   Return the value for property sym of music expression mus. If no value is found, return val or '()' if val is not specified. [Function]

ly:music-set-property! mus sym val
   Set property sym in music expression mus to val. [Function]

ly:music-transpose m p
   Transpose m such that central C is mapped to p. Return m. [Function]

ly:note-column-accidentals note-column
   Return the AccidentalPlacement grob from note-column if any, or SCM_EOL otherwise. [Function]

ly:note-column-dot-column note-column
   Return the DotColumn grob from note-column if any, or SCM_EOL otherwise. [Function]

ly:note-head::stem-attachment font-metric glyph-name
   Get attachment in font-metric for attaching a stem to notehead glyph-name. [Function]

ly:number->string s
   Convert s to a string without generating many decimals. [Function]

ly:one-line-breaking pb
   Put each score on a single line, and put each line on its own page. The paper-width setting will be modified so that every page will be wider than the widest line. [Function]

ly:optimal-breaking pb
   Optimally break (pages and lines) the Paper_book object pb to minimize badness in both vertical and horizontal spacing. [Function]

ly:option-usage port
   Print ly:set-option usage. Optional port argument for the destination defaults to current output port. [Function]

ly:otf->cff otf-file-name
   Convert the contents of an OTF file to a CFF file, returning it as a string. [Function]

ly:otf-font? font
   Is font an OpenType font? [Function]

ly:otf-font-glyph-info font glyph
   Given the font metric font of an OpenType font, return the information about named glyph glyph (a string). [Function]

ly:otf-font-table-data font tag
   Extract a table tag from font. Return empty string for non-existent tag. [Function]

ly:otf-glyph-count font
   Return the number of glyphs in font. [Function]

ly:otf-glyph-list font
   Return a list of glyph names for font. [Function]

ly:output-def? def
   Is def an output definition? [Function]
ly:output-def-clone def
Clone output definition def.

ly:output-def-lookup def sym val
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
Return the parent output definition of def.

ly:output-def-scope def
Return the variable scope inside def.

ly:output-def-set-variable! def sym val
Set an output definition def variable sym to val.

ly:output-description output-def
Return the description of translators in output-def.

ly:output-find-context-def output-def context-name
Return an alist of all context defs (matching context-name if given) in output-def.

ly:output-formats
Formats passed to '--format' as a list of strings, used for the output.

ly:outputter-close outputter
Close port of outputter.

ly:outputter-dump-stencil outputter stencil
Dump stencil expr onto outputter.

ly:outputter-dump-string outputter str
Dump str onto outputter.

ly:outputter-module outputter
Return output module of outputter.

ly:outputter-output-scheme outputter expr
Eval expr in module of outputter.

ly:outputter-port outputter
Return output port for outputter.

ly:page-marker? x
Is x a Page_marker object?

ly:page-turn-breaking pb
Optimally break (pages and lines) the Paper_book object pb such that page turns only happen in specified places, returning its pages.

ly:pango-font? f
Is f a pango font?

ly:pango-font-physical-fonts f
Return alist of (ps-name file-name font-index) lists for Pango font f.

ly:paper-book? x
Is x a Paper_book object?
ly:paper-book-header pb
Return the header definition (\header) in Paper_book object pb.

ly:paper-book-pages pb

ly:paper-book-paper pb
Return the paper output definition (\paper) in Paper_book object pb.

ly:paper-book-performances pb

ly:paper-book-scopes pb

ly:paper-book-systems pb

ly:paper-fonts def
Return a list containing the fonts from output definition def (e.g., \paper).

ly:paper-get-font def chain
Find a font metric in output definition def satisfying the font-qualifiers in alist chain chain, and return it. (An alist chain is a list of alists, containing grob properties.)

ly:paper-get-number def sym
Return the value of variable sym in output definition def as a double.

ly:paper-outputscale def
Return the output-scale for output definition def.

ly:paper-score-paper-systems paper-score
Return vector of paper_system objects from paper-score.

ly:paper-system? obj
Is obj a C++ Prob object of type paper-system?

ly:paper-system-minimum-distance sys1 sys2
Measure the minimum distance between these two paper-systems, using their stored skylines if possible and falling back to their extents otherwise.

ly:parse-file name

ly:parse-string-expression parser-smob ly-code filename line
Parse the string ly-code with parser-smob. Return the contained music expression. filename and line are optional source indicators.

ly:parsed-undead-list!
Return the list of objects that have been found live that should have been dead, and clear that list.

ly:parser-clear-error parser
Clear the error flag for the parser.

ly:parser-clone parser-smob closures location
Return a clone of parser-smob. An association list of port positions to closures can be specified in closures in order to have $ and # interpreted in their original lexical environment. If location is a valid location, it becomes the source of all music expressions inside.
ly:parser-define! parser-smob symbol val
   Bind symbol to val in parser-smob’s module.

ly:parser-error parser msg input
   Display an error message and make the parser fail.

ly:parser-has-error? parser
   Does parser have an error flag?

ly:parser-include-string parser-smob ly-code
   Include the string ly-code into the input stream for parser-smob. Can only be used in immediate Scheme expressions ($ instead of #).

ly:parser-lexer parser-smob
   Return the lexer for parser-smob.

ly:parser-lookup parser-smob symbol
   Look up symbol in parser-smob’s module. Return ’() if not defined.

ly:parser-output-name parser
   Return the base name of the output file.

ly:parser-parse-string parser-smob ly-code
   Parse the string ly-code with parser-smob. Upon failure, throw ly-file-failed key.

ly:parser-set-note-names parser names
   Replace current note names in parser. names is an alist of symbols. This only has effect if the current mode is notes.

ly:performance-write performance filename
   Write performance to filename.

ly:pfb->pfa pfb-file-name
   Convert the contents of a Type 1 font in PFB format to PFA format.

ly:pitch? x
   Is x a Pitch object?

ly:pitch<? p1 p2
   Is p1 lexicographically smaller than p2?

ly:pitch-alteration pp
   Extract the alteration from pitch pp.

ly:pitch-diff pitch root
   Return pitch delta such that pitch transposed by delta equals root.

ly:pitch-negate p
   Negate p.

ly:pitch-notename pp
   Extract the note name from pitch pp.

ly:pitch-octave pp
   Extract the octave from pitch pp.

ly:pitch-quartertones pp
   Calculate the number of quarter tones of pp from middle C.
**ly:pitch-semitones** \( pp \)
Calculate the number of semitones of \( pp \) from middle C.

**ly:pitch-steps** \( p \)
Number of steps counted from middle C of the pitch \( p \).

**ly:pitch-tones** \( pp \)
Calculate the number of tones of \( pp \) from middle C as a rational number.

**ly:pitch-transpose** \( p \ \delta \)
Transpose \( p \) by the amount \( \delta \), where \( \delta \) is relative to middle C.

**ly:pointer-group-interface::add-grob** grob sym grob-element
Add grob-element to grob's sym grob array.

**ly:position-on-line?** sg spos
Return whether \( spos \) is on a line of the staff associated with the grob \( sg \) (even on an extender line).

**ly:prob?** \( x \)
Is \( x \) a Prob object?

**ly:prob-immutable-properties** prob
Retrieve an alist of immutable properties.

**ly:prob.mutable-properties** prob
Retrieve an alist of mutable properties.

**ly:prob-property** prob sym val
Return the value for property \( sym \) of Prob object \( prob \). If no value is found, return \( val \) or \' () if \( val \) is not specified.

**ly:prob-property?** obj sym
Is boolean prop \( sym \) of sym set?

**ly:prob-set-property!** obj sym value
Set property \( sym \) of \( obj \) to \( value \).

**ly:prob-type?** obj type
Is \( obj \) the specified prob-type?

**ly:programming-error** str rest
A Scheme callable function to issue the internal warning \( str \). The message is formatted with \texttt{format} and \texttt{rest}.

**ly:progress** str rest
A Scheme callable function to print progress \( str \). The message is formatted with \texttt{format} and \texttt{rest}.

**ly:property-lookup-stats** sym
Return hash table with a property access corresponding to \( sym \). Choices are \texttt{prob, grob}, and \texttt{context}.

**ly:protects**
Return hash of protected objects.

**ly:pt** \( num \)
\( num \) printer points.
Appendix A: Notation manual tables

ly:register-stencil-expression symbol
Add symbol as head of a stencil expression.

ly:relative-group-extent elements common axis
Determine the extent of elements relative to common in the axis direction.

ly:reset-all-fons
Forget all about previously loaded fonts.

ly:round-filled-box xext yext blot
Make a Stencil object that prints a black box of dimensions xext, yext and roundness blot.

ly:round-filled-polygon points blot
Make a Stencil object that prints a black polygon with corners at the points defined by points (list of coordinate pairs) and roundness blot.

ly:run-translator mus output-def
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 to uniquely identify the score block containing it.

ly:score? x
Is x a Score object?

ly:score-add-output-def! score def
Add an output definition def to score.

ly:score-embedded-format score layout
Run score through layout (an output definition) scaled to correct output-scale already, returning a list of layout-lines.

ly:score-error? score
Was there an error in the score?

ly:score-header score
Return score header.

ly:score-music score
Return score music.

ly:score-output-defs score
All output definitions in a score.

ly:score-set-header! score module
Set the score header.

ly:set-default-scale scale
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
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.
\texttt{ly:set-middle-C! context}

Set the \texttt{middleCPosition} variable in \texttt{context} based on the variables \texttt{middleCClefPosition} and \texttt{middleCOffset}.

\texttt{ly:set-option \texttt{var val}}

Set a program option.

\texttt{ly:set-property-cache-callback \texttt{cb}}

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.

\texttt{ly:simple-closure? \texttt{clos}}

Is \texttt{clos} a simple closure?

\texttt{ly:skyline? \texttt{x}}

Is \texttt{x} a \texttt{Skyline} object?

\texttt{ly:skyline-empty? \texttt{sky}}

Return whether \texttt{sky} is empty.

\texttt{ly:skyline-pair? \texttt{x}}

Is \texttt{x} a \texttt{Skyline_pair} object?

\texttt{ly:slur-score-count}

Count number of slur scores.

\texttt{ly:smob-protects}

Return LilyPond’s internal smob protection list.

\texttt{ly:solve-spring-rod-problem \texttt{springs rods length ragged}}

Solve a spring and rod problem for \texttt{count} objects, that are connected by \texttt{count-1} springs, and an arbitrary number of \texttt{rods}. \texttt{count} is implicitly given by \texttt{springs} and \texttt{rods}. The \texttt{springs} argument has the format \texttt{(ideal, inverse\_hook)} and \texttt{rods} is of the form \texttt{(idx1, idx2, distance)}.

\texttt{length} is a number, \texttt{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 \texttt{spring-count+1} positions of the objects.

\texttt{ly:source-file? \texttt{x}}

Is \texttt{x} a \texttt{Source\_file} object?

\texttt{ly:spanner? \texttt{g}}

Is \texttt{g} a spanner object?

\texttt{ly:spanner-bound \texttt{spanner dir}}

Get one of the bounds of \texttt{spanner}. \texttt{dir} is \texttt{-1} for left, and \texttt{1} for right.

\texttt{ly:spanner-broken-into \texttt{spanner}}

Return broken-into list for \texttt{spanner}.

\texttt{ly:spanner-set-bound! \texttt{spanner dir item}}

Set grob \texttt{item} as bound in direction \texttt{dir} for \texttt{spanner}.
**ly:spawn** command rest  
Simple interface to g_spawn_sync str. The error is formatted with format and rest.

**ly:spawn? x**  
Is x a Spring object?

**ly:spawn-set-inverse-compress-strength! spring strength**  
Set the inverse compress strength of spring.

**ly:spawn-set-inverse-stretch-strength! spring strength**  
Set the inverse stretch strength of spring.

**ly:staff-symbol-line-thickness grob**  
Returns the line-thickness of the staff associated with grob.

**ly:staff-symbol-staff-radius grob**  
Returns the radius of the staff associated with grob.

**ly:staff-symbol-staff-space grob**  
Returns the staff-space of the staff associated with grob.

**ly:start-environment**  
Return the environment (a list of strings) that was in effect at program start.

**ly:stderr-redirect file-name mode**  
Redirect stderr to file-name, opened with mode.

**ly:stencil? x**  
Is x a Stencil object?

**ly:stencil-add args**  
Combine stencils. Takes any number of arguments.

**ly:stencil-aligned-to stil axis dir**  
Align stil using its own extents. dir is a number. -1 and 1 are left and right, respectively. Other values are interpolated (so 0 means the center).

**ly:stencil-combine-at-edge first axis direction second padding**  
Construct a stencil by putting second next to first. axis can be 0 (x-axis) or 1 (y-axis). direction can be -1 (left or down) or 1 (right or up). The stencils are juxtaposed with padding as extra space. first and second may also be ’() or #f.

**ly:stencil-empty? stil axis**  
Return whether stil is empty. If an optional axis is supplied, the emptiness check is restricted to that axis.

**ly:stencil-expr stil**  
Return the expression of stil.

**ly:stencil-extent stil axis**  
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**  
Analyze s, and return a list of fonts used in s.

**ly:stencil-in-color stc r g b**  
Put stc in a different color.
Appendix A: Notation manual tables

ly:stencil-rotate stil angle x y
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
Return a stencil stil rotated angle degrees around point (x, y), given in absolute coordinates.

ly:stencil-scale stil x y
Scale stil using the horizontal and vertical scaling factors x and y.

ly:stencil-stack first axis direction second padding mindist
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
Return a stil, but translated by offset (a pair of numbers).

ly:stencil-translate-axis stil amount axis
Return a copy of stil but translated by amount in axis direction.

ly:stream-event? obj
Is obj a Stream_event object?

ly:string-percent-encode str
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
Replace string a by string b in string s.

ly:system-font-load name
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
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
Translates a string in C++ printf format and modifies it to use it for scheme formatting.

ly:translator? x
Is x a Translator object?
ly:translator-context trans
Return the context of the translator object trans.

ly:translator-description me
Return an alist of properties of translator me.

ly:translator-group? x
Is x a Translator_group object?

ly:translator-name trans
Return the type name of the translator object trans. The name is a symbol.

ly:transpose-key-alist l pit
Make a new key alist of l transposed by pitch pit.

ly:truncatem-list! lst i
Take at most the first i of list lst.

ly:ttf->pfa ttf-file-name idx
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
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:undead? x
Is x a Undead object?

ly:unit
Return the unit used for lengths as a string.

ly:unpure-pure-container? clos
Is clos an unpure pure container?

ly:unpure-pure-container-pure-part pc
Return the pure part of pc.

ly:unpure-pure-container-unpure-part pc
Return the unpure part of pc.

ly:usage
Print usage message.

ly:verbose-output?
Was verbose output requested, i.e. loglevel at least DEBUG?

ly:version
Return the current lilypond version as a list, e.g., (1 3 127 uu1).

ly:warning str rest
A Scheme callable function to issue the warning str. The message is formatted with format and rest.

ly:warning-located location str rest
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

Encode the Unicode codepoint wc, an integer, as UTF-8.
## Appendix B Cheat sheet

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>1 2 8 16</code></td>
<td>durations</td>
<td><img src="example1.png" alt="Example" /></td>
</tr>
<tr>
<td><code>c4. c4..</code></td>
<td>augmentation dots</td>
<td><img src="example2.png" alt="Example" /></td>
</tr>
<tr>
<td><code>c d e f g a b</code></td>
<td>scale</td>
<td><img src="example3.png" alt="Example" /></td>
</tr>
<tr>
<td><code>fis bes</code></td>
<td>alteration</td>
<td><img src="example4.png" alt="Example" /></td>
</tr>
<tr>
<td><code>\clef treble \clef bass</code></td>
<td>clefs</td>
<td><img src="example5.png" alt="Example" /></td>
</tr>
<tr>
<td><code>\time 3/4 \time 4/4</code></td>
<td>time signature</td>
<td><img src="example6.png" alt="Example" /></td>
</tr>
<tr>
<td><code>r4 r8</code></td>
<td>rest</td>
<td><img src="example7.png" alt="Example" /></td>
</tr>
<tr>
<td><code>d ~ d</code></td>
<td>tie</td>
<td><img src="example8.png" alt="Example" /></td>
</tr>
<tr>
<td><code>\key es \major</code></td>
<td>key signature</td>
<td><img src="example9.png" alt="Example" /></td>
</tr>
</tbody>
</table>
\textit{note}' \quad \text{raise octave}

\textit{note}, \quad \text{lower octave}

c(\ d\ e) \quad \text{slur}

c\(\ c( \ d\ e\)\) \quad \text{phrasing slur}

a8[\ b] \quad \text{beam}

<< \texttt{\new Staff \ldots \new Staff} >> \quad \text{more staves}

c-> c-. \quad \text{articulations}

c2\mf\ c\sfz \quad \text{dynamics}

a\!<\ a\ a\)! \quad \text{crescendo}
a\> a a\! decrescendo

< > chord

\partial 8 pickup / upbeat

\tuplet 3/2 \{f g a\} triplets

\grace grace notes

\lyricmode \{ twinkle \} entering lyrics

\new Lyrics printing lyrics
twinkle
twinkle

twin -- kle lyric hyphen

c\dim f:maj7 chords

\chordmode \{ c:d\im f:m\aj7 \} chords

\context ChordNames printing chord names

<<\{ e f\} \ \{ c d\}>> polyphony
s4 s8 s16

spacer rests
Appendix C GNU Free Documentation License

Version 1.3, 3 November 2008


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. Secondarily, 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. 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 pro-
vided 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 copy-
right 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.
Appendix D  LilyPond command index

This index lists all the LilyPond commands and keywords with links to those sections of the manual which describe or discuss their use. Each link is in two parts. The first part points to the exact location in the manual where the command or keyword appears; the second part points to the start of the section of the manual in which the command or keyword appears.

! .................................  6
/: ..................................  6
/: ..................................  86
/: ..................................  86
/; ..................................  1
/; ..................................  1
/- ................................ 111
/- ................................ 111
/- ................................ 111
\abs-fontsize ...................... 224, 645
\accent ................................ 2
\accentual ................................ 2
\accentual ............................. 2
\accentual ................................ 2
\accidentalStyle .................... 25
\addChordShape ..................... 347
\addInstrumentDefinition .......... 195, 202
\addlyrics ......................... 241, 243, 244
\addQuote ................................ 195
\aeolian ................................ 20
\afterGrace ................................ 104
\aikenHeads .......................... 37
\aikenHeadsMinor ................... 38
\alias ................................ 560
\allowPageTurn ........................ 511
\alterBroken .......................... 601
\alternative .......................... 138
\appendToTag .......................... 475
\appoggiatura ....................... 104
\arpeggio ................................ 133
\arpeggioArrowDown .................. 133
\arpeggioArrowUp .................... 133
\arpeggioBracket ..................... 133
\arpeggioNormal ..................... 133
\arpeggioParenthesis ................ 133
\arpeggioParenthesisDashed ......... 133
\arrow-head ......................... 231, 669
\ascends ................................ 420, 426
\autumn ................................ 420, 426
\augmentum ........................... 426
\auto-footnote ........................ 690
\autoBeamDiff ........................ 76
\autoBeamOff .......................... 307
\autoBeamOn ........................... 76
\autochange ........................... 305
\backslashed-digit ........................... 690
\texttt{\textbackslash{parenthesize}} 
\texttt{\textbackslash{partcombine}} 
\texttt{\textbackslash{partcombineApartment}} 
\texttt{\textbackslash{partcombineAutomatic}} 
\texttt{\textbackslash{partcombineChords}} 
\texttt{\textbackslash{partcombineSoloI}} 
\texttt{\textbackslash{partcombineSoloII}} 
\texttt{\textbackslash{partcombineUnisono}} 
\texttt{\textbackslash{partial}} 
\texttt{\textbackslash{path}} 
\texttt{\textbackslash{pattern}} 
\texttt{\textbackslash{pes}} 
\texttt{\textbackslash{phrasingSlurDashed}} 
\texttt{\textbackslash{phrasingSlurDashPattern}} 
\texttt{\textbackslash{phrasingSlurDotted}} 
\texttt{\textbackslash{phrasingSlurDown}} 
\texttt{\textbackslash{phrasingSlurHalfDashed}} 
\texttt{\textbackslash{phrasingSlurHalfSolid}} 
\texttt{\textbackslash{phrasingSlurNeutral}} 
\texttt{\textbackslash{phrasingSlurSolid}} 
\texttt{\textbackslash{phrasingSlurUp}} 
\texttt{\textbackslash{phrygian}} 
\texttt{\textbackslash{pitchedTrill}} 
\texttt{\textbackslash{portato}} 
\texttt{\textbackslash{postscript}} 
\texttt{\textbackslash{powerChords}} 
\texttt{\textbackslash{pp}} 
\texttt{\textbackslash{pppp}} 
\texttt{\textbackslash{ppppp}} 
\texttt{\textbackslash{prall}} 
\texttt{\textbackslash{pralldown}} 
\texttt{\textbackslash{prallmordent}} 
\texttt{\textbackslash{prallprall}} 
\texttt{\textbackslash{prallup}} 
\texttt{\textbackslash{predefinedFretboardsOff}} 
\texttt{\textbackslash{predefinedFretboardsOn}} 
\texttt{\textbackslash{property-recursive}} 
\texttt{\textbackslash{pt}} 
\texttt{\textbackslash{pushToTag}} 
\texttt{\textbackslash{put-adjacent}} 
\texttt{\textbackslash{quilisma}} 
\texttt{\textbackslash{quoteDuring}} 
\texttt{\textbackslash{raise}} 
\texttt{\textbackslash{relative}} 
\texttt{\textbackslash{RemoveEmptyStaves}} 
\texttt{\textbackslash{removeWithTag}} 
\texttt{\textbackslash{repeat}} 
\texttt{\textbackslash{repeat percent}} 
\texttt{\textbackslash{repeat tremolo}} 
\texttt{\textbackslash{repeatTie}} 
\texttt{\textbackslash{replace}} 
\texttt{\textbackslash{rest}} 
\texttt{\textbackslash{rest-by-number}} 
\texttt{\textbackslash{retrograde}} 
\texttt{\textbackslash{reverseturn}} 
\texttt{\textbackslash{revert}} 
\texttt{\textbackslash{revertTimeSignatureSettings}} 
\texttt{\textbackslash{rfz}} 
\texttt{\textbackslash{rheel}} 
\texttt{\textbackslash{right-align}} 
\texttt{\textbackslash{right-brace}} 
\texttt{\textbackslash{right-column}} 
\texttt{\textbackslash{rightHandFinger}} 
\texttt{\textbackslash{roman}} 
\texttt{\textbackslash{rotate}} 
\texttt{\textbackslash{rounded-box}} 
\texttt{\textbackslash{rtoe}} 
\texttt{\textbackslash{sacredHarpHeads}} 
\texttt{\textbackslash{sacredHarpHeadsMinor}} 
\texttt{\textbackslash{sans}} 
\texttt{\textbackslash{scale}} 
\texttt{\textbackslash{scaleDurations}} 
\texttt{\textbackslash{score}} 
\texttt{\textbackslash{segno}} 
\texttt{\textbackslash{semiflat}} 
\texttt{\textbackslash{semiGermanChords}} 
\texttt{\textbackslash{semisharp}} 
\texttt{\textbackslash{sesquiflat}} 
\texttt{\textbackslash{sesquisharp}} 
\texttt{\textbackslash{shiftOff}} 
\texttt{\textbackslash{shiftOn}} 
\texttt{\textbackslash{shiftOnn}} 
\texttt{\textbackslash{shortfermata}} 
\texttt{\textbackslash{showKeySignature}} 
\texttt{\textbackslash{showStaffSwitch}} 
\texttt{\textbackslash{signumcongruentiae}} 
\texttt{\textbackslash{simple}} 
\texttt{\textbackslash{skip}} 
\texttt{\textbackslash{slashed-digit}} 
\texttt{\textbackslash{slashedGrace}} 
\texttt{\textbackslash{slurDashed}} 
\texttt{\textbackslash{slurDashPattern}} 
\texttt{\textbackslash{slurDotted}} 
\texttt{\textbackslash{slurDown}} 
\texttt{\textbackslash{slurHalfDashed}} 
\texttt{\textbackslash{slurHalfSolid}} 
\texttt{\textbackslash{slurNeutral}} 
\texttt{\textbackslash{slurSolid}} 
\texttt{\textbackslash{small}} 
\texttt{\textbackslash{smallCaps}} 
\texttt{\textbackslash{smaller}} 
\texttt{\textbackslash{snappizzicato}} 
\texttt{\textbackslash{sostenutoOff}} 
\texttt{\textbackslash{sostenutoOn}} 
\texttt{\textbackslash{southernHarmonyHeads}} 
\texttt{\textbackslash{southernHarmonyHeadsMinor}} 
\texttt{\textbackslash{sp}} 
\texttt{\textbackslash{spe}} 
\texttt{\textbackslash{staccatissimo}} 
\texttt{\textbackslash{staccato}} 
\texttt{\textbackslash{startGroup}} 
\texttt{\textbackslash{startStaff}} 
\texttt{\textbackslash{startTrillSpan}} 
\texttt{\textbackslash{stdBass}} 
\texttt{\textbackslash{stdBassIV}} 
\texttt{\textbackslash{stdBassV}} 
\texttt{\textbackslash{stdBassVI}} 
\texttt{\textbackslash{stemDown}} 
\texttt{\textbackslash{stemNeutral}} 
\texttt{\textbackslash{stemUp}} 
\texttt{\textbackslash{stencil}}
| \topGroup | \topped | \toppedOn | \toppedStaff | \topTrillSpan | \storePrefixedDiagram | \stringTuning | \stropha | \strut | \sub | \super | \sustainOff | \sustainOn | \tabChordRepeats | \tabFullNotation | \table-of-contents | \tag | \taor | \teeny | \tempo | \tenuto | \text | \textLengthOff | \textLengthOff | \textLengthOn | \textLengthOn | \textSpannerDown | \textSpannerNeutral | \textSpannerUp | \thumb | \tieLyric | \tieDashed | \tieDotted | \tieDown | \tieNeutral | \tieSolid | \tieUp | \time | \tiny | \tocItem | \translate | \translate-scaled | \transparent | \transpose | \transposedCueDuring | \transposition | \treCorde | \triangle | \trill | \tuplet | \tupletDown | \tupletNeutral | \turn | \weak | \type | \typewriter | \unaCorda | \underline | \unfoldRepeats | \unhideNotes | \unset | \upbow | \upmordent | \upprail | \upright | \varcoda | \vcenter | \verbatim | \veryLongFermata | \virga | \virgula | \voiceFourStyle | \voiceNeutralStyle | \voiceOne | \voiceOne \voiceFour | \voiceOneStyle | \voiceThreeStyle | \voiceTwoStyle | \void | \vspace | \walkerHeads | \walkerHeadsMinor | \whiteout | \with | \withColor | \withDimensions | \withLink | \withUrl | \woodwindDiagram | \wordwrap | \wordwrapField | \wordwrapInternal | \wordwrapLines | \wordwrapString | \wordwrapStringInternal |

| \applyMusic | \appendToTag | \annotateSpacing | \alignBelowContext | \alias | \akaiHeads | \akaiHeadsMinor | \alignAboveContext | \alignBelowContext | \allowPageTurn | \allowVoltaHook | \alterBroken | \annotateSpacing | \appendToTag | \applyContext | \applyMusic | \applyOutput |

<p>| \absolute | \accepts | \acciaccatura | \accidentalStyle | \addChordShape | \addInstrumentDefinition | \additionalPitchPrefix | \addQuote | \aeolian | \afterGrace | \akaiHeads | \akaiHeadsMinor | \alias | \alignAboveContext | \alignBelowContext | \allowPageTurn | \allowVoltaHook | \alterBroken | \annotateSpacing | \appendToTag | \applyContext | \applyMusic | \applyOutput |</p>
<table>
<thead>
<tr>
<th>Command</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>appoggiatura</td>
<td>736</td>
</tr>
<tr>
<td>arpeggio</td>
<td>133</td>
</tr>
<tr>
<td>arpeggioArrouDown</td>
<td>133</td>
</tr>
<tr>
<td>arpeggioArrouUp</td>
<td>133</td>
</tr>
<tr>
<td>arpeggioBracket</td>
<td>133</td>
</tr>
<tr>
<td>arpeggioNormal</td>
<td>133</td>
</tr>
<tr>
<td>arpeggioParenthesis</td>
<td>133</td>
</tr>
<tr>
<td>arpeggioParenthesisDashed</td>
<td>133</td>
</tr>
<tr>
<td>arrow-head</td>
<td>231</td>
</tr>
<tr>
<td>assertBeamQuant</td>
<td>736</td>
</tr>
<tr>
<td>assertBeamSlope</td>
<td>736</td>
</tr>
<tr>
<td>aug</td>
<td>386</td>
</tr>
<tr>
<td>auto-first-page-number</td>
<td>503</td>
</tr>
<tr>
<td>autoBeaming</td>
<td>78</td>
</tr>
<tr>
<td>autoBeamOff</td>
<td>545</td>
</tr>
<tr>
<td>autoBeamOn</td>
<td>76</td>
</tr>
<tr>
<td>autoBeamOn</td>
<td>76</td>
</tr>
<tr>
<td>autochange</td>
<td>305, 736</td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>Balloon_engraver</td>
<td>211</td>
</tr>
<tr>
<td>balloonGrobText</td>
<td>211, 736</td>
</tr>
<tr>
<td>balloonLengthOff</td>
<td>211</td>
</tr>
<tr>
<td>balloonLengthOn</td>
<td>211</td>
</tr>
<tr>
<td>balloonText</td>
<td>211, 736</td>
</tr>
<tr>
<td>banjo-c-tuning</td>
<td>361</td>
</tr>
<tr>
<td>banjo-modal-tuning</td>
<td>361</td>
</tr>
<tr>
<td>banjo-open-d-tuning</td>
<td>361</td>
</tr>
<tr>
<td>banjo-open-dm-tuning</td>
<td>361</td>
</tr>
<tr>
<td>bar</td>
<td>90, 96, 736</td>
</tr>
<tr>
<td>barCheckSynchronize</td>
<td>101</td>
</tr>
<tr>
<td>BarNumber</td>
<td>97</td>
</tr>
<tr>
<td>barNumberCheck</td>
<td>101, 736</td>
</tr>
<tr>
<td>barNumberVisibility</td>
<td>97</td>
</tr>
<tr>
<td>baryte</td>
<td>96</td>
</tr>
<tr>
<td>base-shortest-duration</td>
<td>531</td>
</tr>
<tr>
<td>baseMoment</td>
<td>78</td>
</tr>
<tr>
<td>beamExceptions</td>
<td>78</td>
</tr>
<tr>
<td>beatStructure</td>
<td>78</td>
</tr>
<tr>
<td>bendafter</td>
<td>127, 736</td>
</tr>
<tr>
<td>binding-offset</td>
<td>501</td>
</tr>
<tr>
<td>blank-after-score-page-penalty</td>
<td>502</td>
</tr>
<tr>
<td>blank-last-page-penalty</td>
<td>502</td>
</tr>
<tr>
<td>blank-page-penalty</td>
<td>502</td>
</tr>
<tr>
<td>bold</td>
<td>224</td>
</tr>
<tr>
<td>bookOutputName</td>
<td>736</td>
</tr>
<tr>
<td>bookOutputSuffix</td>
<td>736</td>
</tr>
<tr>
<td>bookTitleMarkup</td>
<td>456</td>
</tr>
<tr>
<td>bottom-margin</td>
<td>497</td>
</tr>
<tr>
<td>box</td>
<td>230</td>
</tr>
<tr>
<td>bracket</td>
<td>120, 230, 309</td>
</tr>
<tr>
<td>breakable</td>
<td>76</td>
</tr>
<tr>
<td>breathe</td>
<td>126, 737</td>
</tr>
<tr>
<td>breve</td>
<td>41, 52</td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>cadenzaOff</td>
<td>68</td>
</tr>
<tr>
<td>cadenzaOn</td>
<td>68</td>
</tr>
<tr>
<td>center-align</td>
<td>227</td>
</tr>
<tr>
<td>center-column</td>
<td>228</td>
</tr>
<tr>
<td>change</td>
<td>393</td>
</tr>
<tr>
<td>check-consistency</td>
<td>500</td>
</tr>
<tr>
<td>chordChanges</td>
<td>391</td>
</tr>
<tr>
<td>chordmode</td>
<td>5, 12, 344</td>
</tr>
<tr>
<td>chordNameExceptions</td>
<td>394</td>
</tr>
<tr>
<td>chordNameLowercaseMinor</td>
<td>392</td>
</tr>
<tr>
<td>ChordNames</td>
<td>344</td>
</tr>
<tr>
<td>chordNameSeparator</td>
<td>393</td>
</tr>
<tr>
<td>chordNoteName</td>
<td>393</td>
</tr>
<tr>
<td>chordPrefixSpacer</td>
<td>394</td>
</tr>
<tr>
<td>chordRepeats</td>
<td>737</td>
</tr>
<tr>
<td>chordRootName</td>
<td>392</td>
</tr>
<tr>
<td>circle</td>
<td>230</td>
</tr>
<tr>
<td>clef</td>
<td>16, 737</td>
</tr>
<tr>
<td>color</td>
<td>208</td>
</tr>
<tr>
<td>column</td>
<td>228</td>
</tr>
<tr>
<td>combine</td>
<td>231</td>
</tr>
<tr>
<td>common-shortest-duration</td>
<td>531</td>
</tr>
<tr>
<td>Completion_heads_engraver</td>
<td>72</td>
</tr>
<tr>
<td>Completion_rest_engraver</td>
<td>72</td>
</tr>
<tr>
<td>compoundMeter</td>
<td>737</td>
</tr>
<tr>
<td>compressFullBarRests</td>
<td>56, 57</td>
</tr>
<tr>
<td>consists</td>
<td>560</td>
</tr>
<tr>
<td>controlpitch</td>
<td>9</td>
</tr>
<tr>
<td>cr</td>
<td>114</td>
</tr>
<tr>
<td>cres</td>
<td>115</td>
</tr>
<tr>
<td>crescHairpin</td>
<td>115</td>
</tr>
<tr>
<td>crescTextCresc</td>
<td>115</td>
</tr>
<tr>
<td>cross</td>
<td>34</td>
</tr>
<tr>
<td>crossStaff</td>
<td>737</td>
</tr>
<tr>
<td>cueClef</td>
<td>198, 737</td>
</tr>
<tr>
<td>cueClefUnset</td>
<td>737</td>
</tr>
<tr>
<td>cueDuring</td>
<td>198, 737</td>
</tr>
<tr>
<td>cueDuringWithClef</td>
<td>198, 737</td>
</tr>
<tr>
<td>currentBarNumber</td>
<td>96, 110</td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
<tr>
<td>deadNote</td>
<td>737</td>
</tr>
<tr>
<td>decr</td>
<td>114</td>
</tr>
<tr>
<td>decres</td>
<td>115</td>
</tr>
<tr>
<td>default</td>
<td>25, 27</td>
</tr>
<tr>
<td>default-staff-staff-spacing</td>
<td>515</td>
</tr>
<tr>
<td>defaultBarType</td>
<td>96</td>
</tr>
<tr>
<td>defaultNoteHeads</td>
<td>737</td>
</tr>
<tr>
<td>defaultTimeSignature</td>
<td>60</td>
</tr>
<tr>
<td>defineBarLine</td>
<td>94, 737</td>
</tr>
<tr>
<td>denies</td>
<td>560</td>
</tr>
<tr>
<td>dim</td>
<td>115, 386</td>
</tr>
<tr>
<td>dimHairpin</td>
<td>115</td>
</tr>
<tr>
<td>dimTextDecr</td>
<td>115</td>
</tr>
<tr>
<td>dimTextDecres</td>
<td>115</td>
</tr>
<tr>
<td>dimTextDim</td>
<td>115</td>
</tr>
<tr>
<td>displayLilyMusic</td>
<td>737</td>
</tr>
<tr>
<td>displayMusic</td>
<td>737</td>
</tr>
<tr>
<td>displayScheme</td>
<td>737</td>
</tr>
<tr>
<td>dodecaphonic</td>
<td>30</td>
</tr>
<tr>
<td>dorian</td>
<td>20</td>
</tr>
<tr>
<td>dotsDown</td>
<td>42</td>
</tr>
<tr>
<td>dotsNeutral</td>
<td>42</td>
</tr>
<tr>
<td>dotsUp</td>
<td>42</td>
</tr>
<tr>
<td>draw-circle</td>
<td>231</td>
</tr>
<tr>
<td>draw-line</td>
<td>231</td>
</tr>
<tr>
<td>drummode</td>
<td>175</td>
</tr>
<tr>
<td>Drummode</td>
<td>175</td>
</tr>
<tr>
<td>dynamic</td>
<td>120</td>
</tr>
</tbody>
</table>
Appendix D: LilyPond command index 788

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>dynamicDown</td>
<td>116</td>
</tr>
<tr>
<td>DynamicLineSpanner</td>
<td>116</td>
</tr>
<tr>
<td>dynamicNeutral</td>
<td>116</td>
</tr>
<tr>
<td>dynamicUp</td>
<td>116</td>
</tr>
<tr>
<td>E</td>
<td></td>
</tr>
<tr>
<td>easyHeadsOff</td>
<td>36</td>
</tr>
<tr>
<td>easyHeadsOn</td>
<td>36</td>
</tr>
<tr>
<td>endSpanners</td>
<td>737</td>
</tr>
<tr>
<td>epsfile</td>
<td>232</td>
</tr>
<tr>
<td>espressivo</td>
<td>115</td>
</tr>
<tr>
<td>eventChords</td>
<td>738</td>
</tr>
<tr>
<td>expandFullBarRests</td>
<td>56, 57</td>
</tr>
<tr>
<td>extra-offset</td>
<td>515</td>
</tr>
<tr>
<td>F</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>114</td>
</tr>
<tr>
<td>featherDurations</td>
<td>89</td>
</tr>
<tr>
<td>featherDurations</td>
<td>738</td>
</tr>
<tr>
<td>fermataMarkup</td>
<td>56, 57</td>
</tr>
<tr>
<td>ff</td>
<td>114</td>
</tr>
<tr>
<td>fff</td>
<td>114</td>
</tr>
<tr>
<td>ffff</td>
<td>114</td>
</tr>
<tr>
<td>fill-line</td>
<td>229</td>
</tr>
<tr>
<td>filled-box</td>
<td>231</td>
</tr>
<tr>
<td>finger</td>
<td>205, 738</td>
</tr>
<tr>
<td>first-page-number</td>
<td>503</td>
</tr>
<tr>
<td>followVoice</td>
<td>306</td>
</tr>
<tr>
<td>font-face</td>
<td>205, 236</td>
</tr>
<tr>
<td>font-size</td>
<td>204, 205</td>
</tr>
<tr>
<td>font-size</td>
<td>224</td>
</tr>
<tr>
<td>fontsize</td>
<td>204</td>
</tr>
<tr>
<td>footnote</td>
<td>738</td>
</tr>
<tr>
<td>forget</td>
<td>30</td>
</tr>
<tr>
<td>four-string-banjo</td>
<td>361</td>
</tr>
<tr>
<td>fp</td>
<td>114</td>
</tr>
<tr>
<td>fret-diagram</td>
<td>335</td>
</tr>
<tr>
<td>fret-diagram-interface</td>
<td>340</td>
</tr>
<tr>
<td>fret-diagram-terse</td>
<td>337</td>
</tr>
<tr>
<td>fret-diagram-terse</td>
<td>338</td>
</tr>
<tr>
<td>FretBoards</td>
<td>343</td>
</tr>
<tr>
<td>funkHeads</td>
<td>37</td>
</tr>
<tr>
<td>funkHeadsMinor</td>
<td>98</td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>general-align</td>
<td>228</td>
</tr>
<tr>
<td>glissando</td>
<td>128</td>
</tr>
<tr>
<td>grace</td>
<td>738</td>
</tr>
<tr>
<td>GregorianTranscriptionStaff</td>
<td>175</td>
</tr>
<tr>
<td>Grid_line_span_engraver</td>
<td>212</td>
</tr>
<tr>
<td>Grid_point_engraver</td>
<td>212</td>
</tr>
<tr>
<td>gridInterval</td>
<td>212</td>
</tr>
<tr>
<td>grobsub descriptions</td>
<td>738</td>
</tr>
<tr>
<td>grow-direction</td>
<td>89</td>
</tr>
<tr>
<td>H</td>
<td></td>
</tr>
<tr>
<td>halign</td>
<td>227</td>
</tr>
<tr>
<td>harmonicByFret</td>
<td>738</td>
</tr>
<tr>
<td>harmonicByRatio</td>
<td>738</td>
</tr>
<tr>
<td>harmonicNote</td>
<td>738</td>
</tr>
<tr>
<td>harmonicsOn</td>
<td>738</td>
</tr>
<tr>
<td>hbracket</td>
<td>230</td>
</tr>
<tr>
<td>hide</td>
<td>738</td>
</tr>
<tr>
<td>hideKeySignature</td>
<td>373</td>
</tr>
<tr>
<td>hideNotes</td>
<td>207</td>
</tr>
<tr>
<td>hideStaffSwitch</td>
<td>306</td>
</tr>
<tr>
<td>horizontal-shift</td>
<td>501</td>
</tr>
<tr>
<td>Horizontal_bracket_engraver</td>
<td>214</td>
</tr>
<tr>
<td>huge</td>
<td>204, 226</td>
</tr>
<tr>
<td>I</td>
<td></td>
</tr>
<tr>
<td>improvisationOff</td>
<td>40, 74</td>
</tr>
<tr>
<td>improvisationOn</td>
<td>40, 74</td>
</tr>
<tr>
<td>indent</td>
<td>193, 501, 534</td>
</tr>
<tr>
<td>inner-margin</td>
<td>500</td>
</tr>
<tr>
<td>inStaffSegno</td>
<td>738</td>
</tr>
<tr>
<td>instrumentSwitch</td>
<td>195, 739</td>
</tr>
<tr>
<td>inversion</td>
<td>739</td>
</tr>
<tr>
<td>ionian</td>
<td>20</td>
</tr>
<tr>
<td>italic</td>
<td>224</td>
</tr>
<tr>
<td>J</td>
<td></td>
</tr>
<tr>
<td>justified-lines</td>
<td>235</td>
</tr>
<tr>
<td>justify</td>
<td>229</td>
</tr>
<tr>
<td>K</td>
<td></td>
</tr>
<tr>
<td>keepWithTag</td>
<td>739</td>
</tr>
<tr>
<td>key</td>
<td>20, 38, 739</td>
</tr>
<tr>
<td>killCues</td>
<td>202, 739</td>
</tr>
<tr>
<td>L</td>
<td></td>
</tr>
<tr>
<td>label</td>
<td>739</td>
</tr>
<tr>
<td>laissezVibrer</td>
<td>50</td>
</tr>
<tr>
<td>language</td>
<td>739</td>
</tr>
<tr>
<td>languageRestore</td>
<td>739</td>
</tr>
<tr>
<td>languageSaveAndChange</td>
<td>739</td>
</tr>
<tr>
<td>large</td>
<td>204, 226</td>
</tr>
<tr>
<td>larger</td>
<td>224, 226</td>
</tr>
<tr>
<td>last-bottom-spacing</td>
<td>499</td>
</tr>
<tr>
<td>layout file</td>
<td>506</td>
</tr>
<tr>
<td>left-align</td>
<td>227</td>
</tr>
<tr>
<td>left-margin</td>
<td>499</td>
</tr>
<tr>
<td>line-width</td>
<td>499, 534</td>
</tr>
<tr>
<td>locrian</td>
<td>20</td>
</tr>
<tr>
<td>longa</td>
<td>41, 52</td>
</tr>
<tr>
<td>lover</td>
<td>227</td>
</tr>
<tr>
<td>ly: minimal-breaking</td>
<td>511</td>
</tr>
<tr>
<td>ly: one-line-breaking</td>
<td>511</td>
</tr>
<tr>
<td>ly: optimal-breaking</td>
<td>510</td>
</tr>
<tr>
<td>ly: page-turn-breaking</td>
<td>510</td>
</tr>
<tr>
<td>lydian</td>
<td>20</td>
</tr>
<tr>
<td>M</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>386</td>
</tr>
<tr>
<td>magnify</td>
<td>224</td>
</tr>
<tr>
<td>magstep</td>
<td>204, 578</td>
</tr>
<tr>
<td>maj</td>
<td>386</td>
</tr>
<tr>
<td>major</td>
<td>20</td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>major seven symbols</td>
<td>394</td>
</tr>
<tr>
<td>majorSevenSymbol</td>
<td>393</td>
</tr>
<tr>
<td>make-dynamic-script</td>
<td>120</td>
</tr>
<tr>
<td>make-pango-font-tree</td>
<td>238</td>
</tr>
<tr>
<td>makeClusters</td>
<td>158</td>
</tr>
<tr>
<td>makeDefaultStringTuning</td>
<td>739</td>
</tr>
<tr>
<td>mark</td>
<td>101, 219, 739</td>
</tr>
<tr>
<td>markLengthOff</td>
<td>65, 220</td>
</tr>
<tr>
<td>markLengthOn</td>
<td>65, 220</td>
</tr>
<tr>
<td>markup</td>
<td>219, 221, 222, 223</td>
</tr>
<tr>
<td>markup-markup-spacing</td>
<td>499</td>
</tr>
<tr>
<td>markup-system-spacing</td>
<td>498</td>
</tr>
<tr>
<td>markuplist</td>
<td>222, 235</td>
</tr>
<tr>
<td>max-systems-per-page</td>
<td>501</td>
</tr>
<tr>
<td>maxima</td>
<td>41, 52</td>
</tr>
<tr>
<td>measureLength</td>
<td>78, 110</td>
</tr>
<tr>
<td>measurePosition</td>
<td>67, 110</td>
</tr>
<tr>
<td>MensuralStaff</td>
<td>175</td>
</tr>
<tr>
<td>mergeDifferentlyDottedOff</td>
<td>162</td>
</tr>
<tr>
<td>mergeDifferentlyDottedOn</td>
<td>162</td>
</tr>
<tr>
<td>mergeDifferentlyHeadedOff</td>
<td>162</td>
</tr>
<tr>
<td>mergeDifferentlyHeadedOn</td>
<td>162</td>
</tr>
<tr>
<td>mf</td>
<td>114</td>
</tr>
<tr>
<td>min-systems-per-page</td>
<td>501</td>
</tr>
<tr>
<td>minimum-Y-extent</td>
<td>515</td>
</tr>
<tr>
<td>minimumFret</td>
<td>319, 355</td>
</tr>
<tr>
<td>minimumPageTurnLength</td>
<td>510</td>
</tr>
<tr>
<td>minimumRepeatLengthForPageTurn</td>
<td>511</td>
</tr>
<tr>
<td>minor</td>
<td>20</td>
</tr>
<tr>
<td>minorChordModifier</td>
<td>394</td>
</tr>
<tr>
<td>mixed</td>
<td>309</td>
</tr>
<tr>
<td>mixolydian</td>
<td>20</td>
</tr>
<tr>
<td>modalInversion</td>
<td>15, 739</td>
</tr>
<tr>
<td>modalTranspose</td>
<td>14, 739</td>
</tr>
<tr>
<td>modern</td>
<td>27</td>
</tr>
<tr>
<td>modern-cautionary</td>
<td>28</td>
</tr>
<tr>
<td>modern-voice</td>
<td>28</td>
</tr>
<tr>
<td>modern-voice-cautionary</td>
<td>28</td>
</tr>
<tr>
<td>mp</td>
<td>114</td>
</tr>
<tr>
<td>MultiMeasureRestText</td>
<td>56</td>
</tr>
<tr>
<td>musicGlyph</td>
<td>103</td>
</tr>
<tr>
<td>musicMap</td>
<td>739</td>
</tr>
</tbody>
</table>

**O**

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>octaveCheck</td>
<td>9, 739</td>
</tr>
<tr>
<td>offset</td>
<td>739</td>
</tr>
<tr>
<td>omit</td>
<td>740</td>
</tr>
<tr>
<td>once</td>
<td>740</td>
</tr>
<tr>
<td>oneVoice</td>
<td>159</td>
</tr>
<tr>
<td>ottava</td>
<td>22, 740</td>
</tr>
<tr>
<td>outer-margin</td>
<td>500</td>
</tr>
<tr>
<td>outside-staff-horizontal-padding</td>
<td>529</td>
</tr>
<tr>
<td>outside-staff-padding</td>
<td>529</td>
</tr>
<tr>
<td>outside-staff-priority</td>
<td>529</td>
</tr>
<tr>
<td>overrideProperty</td>
<td>740</td>
</tr>
<tr>
<td>overrideTimeSignatureSettings</td>
<td>740</td>
</tr>
</tbody>
</table>

**P**

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>114</td>
</tr>
<tr>
<td>pad-around</td>
<td>231</td>
</tr>
<tr>
<td>pad-markup</td>
<td>231</td>
</tr>
<tr>
<td>pad-to-box</td>
<td>231</td>
</tr>
<tr>
<td>pad-x</td>
<td>231</td>
</tr>
<tr>
<td>page-breaking</td>
<td>502</td>
</tr>
<tr>
<td>page-breaking-system-system-spacing</td>
<td>502</td>
</tr>
<tr>
<td>page-count</td>
<td>502</td>
</tr>
<tr>
<td>page-spacing-weight</td>
<td>503</td>
</tr>
<tr>
<td>pageBreak</td>
<td>740</td>
</tr>
<tr>
<td>pageTurn</td>
<td>740</td>
</tr>
<tr>
<td>palmMute</td>
<td>740</td>
</tr>
<tr>
<td>palmMuteOn</td>
<td>740</td>
</tr>
<tr>
<td>paper-height</td>
<td>497</td>
</tr>
<tr>
<td>paper-width</td>
<td>499</td>
</tr>
<tr>
<td>parallelMusic</td>
<td>172, 740</td>
</tr>
<tr>
<td>parenthesesize</td>
<td>209, 740</td>
</tr>
<tr>
<td>partcombine</td>
<td>167, 740</td>
</tr>
<tr>
<td>partcombineApart</td>
<td>168</td>
</tr>
<tr>
<td>partcombineAutomatic</td>
<td>168</td>
</tr>
<tr>
<td>partcombineChords</td>
<td>168</td>
</tr>
<tr>
<td>partcombineDown</td>
<td>741</td>
</tr>
<tr>
<td>partcombineForce</td>
<td>741</td>
</tr>
<tr>
<td>partcombineSoloI</td>
<td>168</td>
</tr>
<tr>
<td>partcombineSoloII</td>
<td>168</td>
</tr>
<tr>
<td>partcombineUnisono</td>
<td>168</td>
</tr>
<tr>
<td>partcombineUp</td>
<td>741</td>
</tr>
<tr>
<td>partial</td>
<td>67, 741</td>
</tr>
<tr>
<td>pedalSustainStyle</td>
<td>309</td>
</tr>
<tr>
<td>percent</td>
<td>150</td>
</tr>
<tr>
<td>phrasingSlurDashed</td>
<td>124</td>
</tr>
<tr>
<td>phrasingSlurDashPattern</td>
<td>125, 741</td>
</tr>
<tr>
<td>phrasingSlurDotted</td>
<td>124</td>
</tr>
<tr>
<td>phrasingSlurDown</td>
<td>124</td>
</tr>
<tr>
<td>phrasingSlurHalfDashed</td>
<td>125</td>
</tr>
<tr>
<td>phrasingSlurHalfSolid</td>
<td>125</td>
</tr>
<tr>
<td>phrasingSlurNeutral</td>
<td>124</td>
</tr>
<tr>
<td>phrasingSlurSolid</td>
<td>124</td>
</tr>
<tr>
<td>phrasingSlurUp</td>
<td>124</td>
</tr>
<tr>
<td>phrygian</td>
<td>20</td>
</tr>
<tr>
<td>piano</td>
<td>28</td>
</tr>
<tr>
<td>piano-cautionary</td>
<td>29</td>
</tr>
<tr>
<td>PianoStaff</td>
<td>302, 305</td>
</tr>
<tr>
<td>Pitch_squash_engraver</td>
<td>74</td>
</tr>
<tr>
<td>pitchedTrill</td>
<td>136, 741</td>
</tr>
<tr>
<td>pointAndClickOff</td>
<td>741</td>
</tr>
<tr>
<td>pointAndClickOn</td>
<td>741</td>
</tr>
<tr>
<td>pointAndClickOn</td>
<td>741</td>
</tr>
</tbody>
</table>
Appendix D: LilyPond command index

postscript ........................................... 232
powerChords ...................................... 359
pp ................................................. 114
ppp ................................................. 114
pppp ............................................... 114
predefinedFretboardsOff ......................... 353
predefinedFretboardsOn ......................... 353
print-all-headers .................................. 503
print-first-page-number ......................... 503
print-page-number ................................ 503
pushToTag ........................................ 741

Q
quotedCueEventTypes .............................. 197
quotedEventTypes .................................. 197
quoteDuring ....................................... 195, 198, 741

R
r ................................................. 52
R ................................................. 55
ragged-bottom .................................... 497
ragged-last ....................................... 500, 534
ragged-last-bottom ............................... 497
ragged-right ..................................... 500, 534
raise ............................................. 227
relative ......................................... 2, 5, 12, 305, 741
removeWithTag ................................... 741
repeatCommands ................................... 145
repeatTie ........................................ 50
resetRelativeOctave .............................. 741
rest .............................................. 52
restrainOpenStrings ............................. 319
retrograde ....................................... 134, 741
revertTimeSignatureSettings .................... 741
rfz .............................................. 114
rgb-color ........................................ 209
RhythmicStaff .................................... 175
right-align ....................................... 227
right-margin ..................................... 500
rightHandFinger .................................. 356, 742
rounded-box ...................................... 230

S
s .................................................. 54
sacredHarpHeadsMinor ............................ 38
scaleDurations .................................... 48, 69, 742
score-markup-spacing ............................ 498
score-system-spacing ............................ 499
scoreTitleMarkup ................................ 456
self-alignment-X .................................. 515
set ................................................ 78
set-octavation ................................... 22
settingsFrom ..................................... 742
sf .................................................. 114
sff ............................................... 114
sfz .............................................. 114
shape ............................................. 742
shiftDurations ................................... 742
shiftOff .......................................... 162
shiftOn .......................................... 162
shiftOnn ......................................... 162
short-indent ................................. 193, 501
show-available-fonts ......................... 238
showFirstLength ............................... 481
showKeySignature .............................. 373
showLastLength ................................ 481
showStaffSwitch ................................ 306
single ........................................... 742
skip ............................................. 54, 742
skipTypesetting .................................. 481
slashChordSeparator ....................... 393
slashedGrace .................................... 742
slurDashed ....................................... 122
slurDashPattern ............................... 122, 742
slurDotted ....................................... 122
slurDown ......................................... 122
slurHalfDashed ................................. 122
slurHalfSolid .................................. 122
slurNeutral ...................................... 122
slurSolid ........................................ 122
slurUp .......................................... 123
small ........................................... 204, 226
smaller ......................................... 224, 226
sostenutoOff .................................... 309
sostenutoOn ...................................... 309
southernHarmonyHeads ....................... 37
southernHarmonyHeadsMinor .................. 38
sp ................................................ 114
spacing .......................................... 531
spacingTweaks ................................... 742
Span\_stem\_engraver ........................... 307
span ............................................. 114
staff-affinity .................................... 515
staff-staff-spacing ......................... 515
staff.midiInstrument ......................... 486
Staff\_symbol\_engraver ......................... 189
staffGroup-staff-spacing .................... 115
start-repeat ..................................... 145
startGroup ...................................... 214
startStaff ....................................... 182, 185
startTrillSpan .................................. 135
stem ............................................. 307
stem-spacing-correction .................. 531
stemDown ....................................... 210
stemLeftBeamCount ............................ 87
stemNeutral ..................................... 210
stemRightBeamCount ........................... 87
stemUp .......................................... 210
stopGroup ....................................... 214
stopStaff ........................................ 182, 185, 189
stopTrillSpan .................................... 135
storePredefinedDiagram .................. 347, 742
stringTuning .................................... 331, 742
stringTunings .................................. 331, 343
styledNoteHeads ............................... 742
sub ............................................ 225
suggestAccidentals ......................... 414
super .......................................... 225
sus ............................................. 388
sustainOff ...................................... 309
sustainOn ...................................... 309
system-count ................................... 501
system-separator-markup .................... 503
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>system-system-spacing</td>
<td>499</td>
</tr>
<tr>
<td>systems-per-page</td>
<td>501</td>
</tr>
<tr>
<td>T</td>
<td></td>
</tr>
<tr>
<td>tabChordRepeats</td>
<td>742</td>
</tr>
<tr>
<td>tabChordRepetition</td>
<td>742</td>
</tr>
<tr>
<td>TabStaff</td>
<td>175, 318</td>
</tr>
<tr>
<td>TabVoice</td>
<td>318</td>
</tr>
<tr>
<td>tag</td>
<td>743</td>
</tr>
<tr>
<td>taor</td>
<td>373</td>
</tr>
<tr>
<td>teaching</td>
<td>30</td>
</tr>
<tr>
<td>teeny</td>
<td>204, 226</td>
</tr>
<tr>
<td>tempo</td>
<td>64</td>
</tr>
<tr>
<td>temporary</td>
<td>743</td>
</tr>
<tr>
<td>text</td>
<td>309</td>
</tr>
<tr>
<td>textLength0ff</td>
<td>57</td>
</tr>
<tr>
<td>textLength0ff</td>
<td>217</td>
</tr>
<tr>
<td>textLengthOn</td>
<td>57</td>
</tr>
<tr>
<td>textLengthOn</td>
<td>217</td>
</tr>
<tr>
<td>textSpannerDown</td>
<td>217</td>
</tr>
<tr>
<td>textSpannerNeutral</td>
<td>217</td>
</tr>
<tr>
<td>textSpannerUp</td>
<td>217</td>
</tr>
<tr>
<td>thumb</td>
<td>205</td>
</tr>
<tr>
<td>tieDashed</td>
<td>50</td>
</tr>
<tr>
<td>tieDashPattern</td>
<td>743</td>
</tr>
<tr>
<td>tieDotted</td>
<td>50</td>
</tr>
<tr>
<td>tieDown</td>
<td>50</td>
</tr>
<tr>
<td>tieNeutral</td>
<td>50</td>
</tr>
<tr>
<td>tieSolid</td>
<td>50</td>
</tr>
<tr>
<td>tieUp</td>
<td>50</td>
</tr>
<tr>
<td>time</td>
<td>59, 78, 743</td>
</tr>
<tr>
<td>times</td>
<td>743</td>
</tr>
<tr>
<td>timeSignatureFraction</td>
<td>69</td>
</tr>
<tr>
<td>tiny</td>
<td>204, 226</td>
</tr>
<tr>
<td>tocItem</td>
<td>743</td>
</tr>
<tr>
<td>top-margin</td>
<td>497</td>
</tr>
<tr>
<td>top-markup-spacing</td>
<td>499</td>
</tr>
<tr>
<td>top-system-spacing</td>
<td>499</td>
</tr>
<tr>
<td>translate</td>
<td>228</td>
</tr>
<tr>
<td>translate-scaled</td>
<td>228</td>
</tr>
<tr>
<td>transpose</td>
<td>5, 10, 12, 743</td>
</tr>
<tr>
<td>transposedCueDuring</td>
<td>201, 743</td>
</tr>
<tr>
<td>transposition</td>
<td>24, 195, 743</td>
</tr>
<tr>
<td>treCorde</td>
<td>309</td>
</tr>
<tr>
<td>tremolo</td>
<td>152</td>
</tr>
<tr>
<td>tremoloFlags</td>
<td>152</td>
</tr>
<tr>
<td>triangle</td>
<td>231</td>
</tr>
<tr>
<td>trill</td>
<td>135</td>
</tr>
<tr>
<td>tuplet</td>
<td>44, 69, 743</td>
</tr>
<tr>
<td>tupletDown</td>
<td>44</td>
</tr>
<tr>
<td>tupletNeutral</td>
<td>44</td>
</tr>
<tr>
<td>TupletNumber</td>
<td>45</td>
</tr>
<tr>
<td>tupletNumberFormatFunction</td>
<td>44</td>
</tr>
<tr>
<td>tupletSpan</td>
<td>743</td>
</tr>
<tr>
<td>tupletSpannerDuration</td>
<td>44</td>
</tr>
<tr>
<td>tupletUp</td>
<td>44</td>
</tr>
<tr>
<td>tweak</td>
<td>743</td>
</tr>
<tr>
<td>two-sided</td>
<td>500</td>
</tr>
<tr>
<td>type</td>
<td>560</td>
</tr>
<tr>
<td>U</td>
<td></td>
</tr>
<tr>
<td>unaCorda</td>
<td>309</td>
</tr>
<tr>
<td>underline</td>
<td>224</td>
</tr>
<tr>
<td>undo</td>
<td>744</td>
</tr>
<tr>
<td>unfold</td>
<td>148</td>
</tr>
<tr>
<td>unfoldRepeats</td>
<td>744</td>
</tr>
<tr>
<td>unHideNotes</td>
<td>207</td>
</tr>
<tr>
<td>V</td>
<td></td>
</tr>
<tr>
<td>VaticanaStaff</td>
<td>175</td>
</tr>
<tr>
<td>VerticalAxisGroup</td>
<td>515</td>
</tr>
<tr>
<td>voice</td>
<td>25, 27</td>
</tr>
<tr>
<td>Voice</td>
<td>159</td>
</tr>
<tr>
<td>voiceOne</td>
<td>159</td>
</tr>
<tr>
<td>void</td>
<td>744</td>
</tr>
<tr>
<td>W</td>
<td></td>
</tr>
<tr>
<td>walkerHeads</td>
<td>37</td>
</tr>
<tr>
<td>walkerHeadsMinor</td>
<td>38</td>
</tr>
<tr>
<td>whichBar</td>
<td>96</td>
</tr>
<tr>
<td>with-color</td>
<td>208</td>
</tr>
<tr>
<td>withMusicProperty</td>
<td>744</td>
</tr>
<tr>
<td>wordwrap</td>
<td>229</td>
</tr>
<tr>
<td>wordwrap-lines</td>
<td>235</td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X-offset</td>
<td>515</td>
</tr>
<tr>
<td>x11-color</td>
<td>208, 209</td>
</tr>
<tr>
<td>xNote</td>
<td>744</td>
</tr>
<tr>
<td>xNotesOn</td>
<td>744</td>
</tr>
</tbody>
</table>
Appendix E LilyPond index

In addition to all the LilyPond commands and keywords, this index lists musical terms and words which relate to each of them, with links to those sections of the manual which describe or discuss that topic. Each link is in two parts. The first part points to the exact location in the manual where the topic appears; the second part points to the start of the section of the manual where that topic is discussed.

! ......................................................... 6 ] ......................................................... 86
" ......................................................... 101 ~
"[" ......................................................... 388 ,
', ......................................................... 1 -
,......................................................... 111 .
. ......................................................... 42 /
/ ......................................................... 388 /* ......................................................... 389 :
: ......................................................... 152 <
< ......................................................... 154 <...> ......................................................... 154 =
= ......................................................... 9 >
> ......................................................... 154 ?
? ......................................................... 6 [
[ ......................................................... 86
Appendix E: LilyPond index

\stemDown .................................................. 210
\stemNeutral .............................................. 210
\stemUp .................................................... 210
\scale ....................................................... 695
\stencil ..................................................... 214
\stop .......................................................... 111
\stopGroup .................................................. 182, 185, 189
\stopTrillSpan .............................................. 135
\storePredefinedDiagram .................................. 347
\stringTuning .............................................. 331
\stropha ..................................................... 420, 426
\strut ......................................................... 695
\sub .......................................................... 225, 652
\super ......................................................... 225, 652
\sustainOff ................................................ 309
\sustainOn .................................................. 309
\tabChordRepeats .......................................... 320
\tabFullNotation .......................................... 318
\table-of-contents ...................................... 472, 697
\tag ............................................................ 475
\tao ........................................................... 373
\teeny ........................................................ 204, 226, 653
\tempo ........................................................ 64
\tenuto ......................................................... 111
\text .......................................................... 653
\textLengthDiff ............................................ 57
\textLengthDiff ............................................ 217
\textLengthOn ............................................... 57
\textLengthOn .............................................. 217
\textSpannerDown .......................................... 217
\textSpannerNeutral ....................................... 217
\textSpannerUp ............................................. 217
\thumb ......................................................... 111, 205
\tied-lyric .................................................. 681
\tieDashed .................................................. 50
\tieDotted ................................................... 50
\tieNeutral .................................................. 50
\tieSolid ..................................................... 50
\tieUp .......................................................... 50
\time .......................................................... 59, 78
\tiny ............................................................ 204, 226, 653
\tocItem ....................................................... 472
\translate ................................................... 228, 666
\translate-scaled ......................................... 228, 666
\transparent ................................................ 695
\transpose .................................................... 5, 10, 12
\transposeViaVoice .......................................... 201
\transposition ............................................. 24, 195
\treCorde .................................................... 309
\triangle ....................................................... 231, 676
\trl ............................................................ 111, 135
\uplet ........................................................ 44, 69
\upletDown .................................................. 44
\upletNeutral .............................................. 44
\upletUp ...................................................... 44
\turn .......................................................... 111
\tweet ........................................................ 571, 573
\type .......................................................... 560
\typewriter .................................................. 653
\unaCorda .................................................... 309
\underline .................................................... 224, 653
\unfoldRepeats .............................................. 487
\unHideNotes ................................................. 207
\unset ........................................................ 568
\upbow ......................................................... 111, 313
\upmordent .................................................. 111
\upprall ...................................................... 111
\upright ...................................................... 654
\varcoda ....................................................... 111
\vcenter ...................................................... 666
\verbatim-file .............................................. 695
\verylongfermata ......................................... 111
\virga ........................................................ 420, 426
\virgula ....................................................... 418
\voiceFourStyle ............................................ 162
\voiceNeutralStyle ......................................... 162
\voiceOne ..................................................... 159
\voiceOne ... \voiceFour .................................. 159
\voiceOneStyle ............................................. 162
\voiceThreeStyle ........................................... 162
\voiceTwoStyle ............................................. 162
\void .......................................................... 493
\vpEase ....................................................... 666
\walkerHeads .............................................. 37
\walkerHeadsMinor ......................................... 38
\whiteout ..................................................... 695
\with ........................................................ 553, 558
\with-color .................................................. 208, 695
\with-dimensions .......................................... 696
\with-link ................................................... 696
\with-url .................................................... 676
\woodwind-diagram ......................................... 684
\wordwrap .................................................... 228, 667
\wordwrap-field ............................................ 667
\wordwrap-internal ......................................... 697
\wordwrap-lines ............................................ 235, 697
\wordwrap-string .......................................... 668
\wordwrap-string-internal .................................. 697

| ..................................................................... 101
~ ..................................................................... 49

1

15ma ............................................................. 22

8

8va ............................................................. 22
8ve ............................................................. 22

A

a due part ....................................................... 167
absolute ......................................................... 1
absolute ......................................................... 2, 735
absolute dynamics ......................................... 114
absolute octave entry ....................................... 1
absolute octave specification ................................ 1
accelerando in MIDI .......................................... 486
accent .......................................................... 112, 699
accentus ......................................................... 699
accepts ........................................................ 560
Appendix E: LilyPond index

acciaccatura ...................................................... 104
acciaccatura .................................................. 735
acciaccatura, multi-note .................................... 108
acciaccatura .................................................... 5
acciaccatura on tied note .................................... 6
acciaccatura style ............................................. 25
acciaccatura style, cautionary, modern voice .......... 28
acciaccatura style, default .................................. 25, 27
acciaccatura style, forget .................................... 30
acciaccatura style, modern ................................... 27, 28
acciaccatura style, modern voice cautionary .......... 28
acciaccatura style, modern-cautionary .................. 27
acciaccatura style, neo-modern ................................ 29
acciaccatura style, neo-modern-cautionary ............ 29
acciaccatura style, neo-modern voice .................... 29
acciaccatura style, neo-modern-voice-cautionary ..... 30
acciaccatura style, no reset .................................. 30
acciaccatura style, piano ..................................... 28
acciaccatura style, piano cautionary .................... 29
acciaccatura style, teaching ................................ 30
acciaccatura style, voice .................................... 27
acciaccatura style, voice, modern cautionary .......... 28
acciaccatura, cautionary ..................................... 6
Accidental, musica ficta ...................................... 414
accidental, parenthesized .................................... 6
accidental, quarter-tone ..................................... 7
accidental, reminder .......................................... 7
accidentals ....................................................... 25, 413, 417, 428
accidentals and simultaneous notes ....................... 31
accidentals in chords ........................................ 31
accidentals, automatic ....................................... 25
accidentals, cadenzas ........................................ 68
accidentals, modern .......................................... 28
accidentals, modern cautionary style .................... 28
accidentals, modern style .................................... 27
accidentals, multivoice ...................................... 28
accidentals, piano ............................................ 28
accidentals, piano cautionary .............................. 29
accidentals, unmetered music ............................... 68
acciaccaturaStyle .............................................. 735
accordion ......................................................... 309
accordion discant symbols .................................. 310
accordion shift symbols ..................................... 310
accordion shifts ............................................... 310
addChordShape ................................................. 347, 735
adding a white background to text ......................... 695
adding custom fret diagrams ................................ 346
addInstrumentDefinition ..................................... 195, 202, 735
additionalPitchPrefix ....................................... 393
additions, in chords ......................................... 387
addQuote ......................................................... 195, 735
adjusting staff symbol ...................................... 578
aeolian ............................................................. 20
afterGrace ....................................................... 104, 736
afterGraceFraction .......................................... 704
Aiken shape note heads ...................................... 37
aikenHeads ...................................................... 37
aikenHeadsMinor ................................................. 38
al niente ............................................................. 117
alias ................................................................. 560
align to objects ............................................... 594
alignAboveContext ........................................... 563
alignBelowContext ........................................... 202, 563
aligning markup text ...................................... 227
aligning markups ............................................. 227
aligning text .................................................... 227
aligning to cadenza .......................................... 109
alignment, text, commands .................................. 230
alist ................................................................. 702
allowPageTurn .................................................. 736
allowVoltaHook ............................................... 736
alterBroken ..................................................... 736
altered chords .................................................. 387
alternate endings ............................................. 138
alternate endings, repeats .................................. 148
alternate repeats .............................................. 148
alternative endings and lyrics .............................. 263
alternative endings, with ties ............................ 141
alternative melody, switching to ......................... 273
alto clef ............................................................. 16
Amazing Grace bagpipe example ......................... 374
ambitus ............................................................. 32
anacrusis .......................................................... 67
anacrusis in a repeat ......................................... 140
analysis, musicological ..................................... 214
ancient clef ...................................................... 16
angled hairpin ................................................... 590
annotate-spacing .............................................. 541
anthems .......................................................... 278
appendToTag ................................................... 736
applyContext .................................................. 736
applyMusic ....................................................... 736
applyOutput ..................................................... 736
appoggiation .................................................... 104
appoggiatura ..................................................... 736
Arabic key signatures ........................................ 437
Arabic music .................................................... 435
Arabic music example ........................................ 439
Arabic music template ....................................... 439
Arabic note names ............................................ 436
Arabic semi-flat symbol ..................................... 436
Arabic time signatures ....................................... 438
arpeggio ........................................................... 133
arpeggio symbols, special .................................. 133
arpeggio, cross-staff parenthesis-style .................... 135
arpeggio, parenthesis-style, cross-staff ................. 135
arpeggioArrowDown .......................................... 133
arpeggioArrowUp .............................................. 133
arpeggioBracket .............................................. 133
arpeggioNormal ............................................... 133
arpeggioParenthesis ......................................... 133
arpeggioParenthesisDashed ................................ 133
arrow-head ...................................................... 231
articulate scripts .............................................. 486
articulation-event ............................................ 197
articulations ................................................... 111, 418
articulations in MIDI ......................................... 486
artificial harmonics ......................................... 314
assertBeamQuant .............................................. 736
assertBeamSlope ............................................... 736
associatedVoice .............................................. 241
associatedVoice .............................................. 243, 273
association list ............................................... 702
aug ................................................................. 386
auto-first-page-number ...................................... 503
autoBeaming ................................................... 78
autoBeaming ................................................... 545
autoBeaming properties for time signatures ............ 60
autoBeamOff ..................................................... 76
autoBeamOn ...................................................... 76
broken chord .............................................. 133
broken spanners, modifying ...................... 601

C
C clef ................................................. 16
cadenza ............................................. 68, 109
cadenza, accidentals ............................... 68
cadenza, aligning to ................................. 109
cadenza, bar lines .................................. 68
cadenza, bar numbers ............................... 68
cadenza, beams ...................................... 68
cadenza, line breaks ................................. 69
cadenza, page breaks ............................... 69
cadenzaOff .......................................... 68
cadenzaOn ........................................... 68
caeusura ............................................. 127
callback .............................................. 702
capo ................................................... 358
cautory accidental ................................... 6
cautory accidental style, piano .................. 29
cautory accidentals, piano ......................... 29
center-align .......................................... 227
center-column ........................................ 228
centered dynamics in piano music ............... 302
centering a column of text ....................... 654
centering text on the page ......................... 229
change .................................................. 303
changing direction of text columns ............... 656
changing fonts ....................................... 224
changing instrument names ....................... 194
changing properties ............................... 567
changing staff automatically ..................... 305
changing staff manually ............................ 303
chants ................................................... 289
character names ..................................... 282
check-consistency ................................... 500
choir staff ........................................... 176
choral tenor clef .................................... 17
choral chords ........................................ 384
choral diagrams ...................................... 334, 343
choral diagrams, automatic ....................... 353
choral glissandi ....................................... 329
choral inversions ................................... 388
choral mode .......................................... 384
choral names .......................................... 384, 389
Chord names in MIDI ................................. 486
chord names with fret diagrams ................. 344
chord quality ......................................... 385
chord shapes for fretted instruments ............ 347
chord steps, altering ............................... 388
chord, broken ......................................... 133
chord, modifying one note in .................... 571
Chord, repetition .................................... 156, 320
chordChanges ........................................ 391
chordMode ............................................ 5, 12, 344
chordNameExceptions ............................... 394
chordNameLowercaseMinor ......................... 392
ChordNames .......................................... 344
chordNameSeparator ................................ 393
chordNoteName .................................... 393
chordPrefixSpacer .................................. 394
chordRepeats ....................................... 747
chordRootName ..................................... 392

chords ................................................. 154, 389
chords and relative octave entry .................. 4
chords and ties ....................................... 49
chords, accidentals in ................................ 31
chords, cross-staff .................................. 307
chords, empty ......................................... 155
chords, fingering ..................................... 205
chords, jazz ........................................... 392
chords, power ......................................... 359
chords, relative pitch ................................ 155
chords, splitting across staves with \autochange
....................................................... 306
chords, suppressing repeated ...................... 391
Christian Harmony note heads .................... 37
church modes .......................................... 20
church rest ............................................ 57
circle .................................................. 230
circling text .......................................... 669
circulus ................................................. 699
clef ..................................................... 5
clef, alto ............................................. 16
clef, ancient .......................................... 16
clef, baritone ......................................... 16
clef, bass ............................................. 16
clef, C ............................................... 16
clef, F ............................................... 16
clef, french .......................................... 16
clef, G ................................................. 16
clef, mezzosoprano .................................. 16
clef, moderntab ....................................... 333
clef, percussion ....................................... 362
clef, soprano ......................................... 16
clef, sub bass ......................................... 16
clef, tab ............................................... 333
clef, tenor ............................................ 16
clef, transposing ..................................... 17
clef, treble .......................................... 16
clef, varbaritone ..................................... 16
clef, violin ........................................... 16
clef, visibility following explicit change ....... 587
clefs .................................................... 409, 417, 427
clefs, visibility of transposition ................. 589
closing bar lines ...................................... 90
closure .................................................. 702
cluster .................................................. 158
coda ..................................................... 103, 111, 699
coda on bar line ...................................... 219
collision, bar number ................................ 100
collisions .............................................. 162
collisions, clashing note columns ............... 158
collisions, cross-staff voices ..................... 304
collisions, ignoring .................................. 158, 167
color .................................................... 208
color in chords ....................................... 209
color, rgb ............................................. 209
colored notes ......................................... 208
colored notes in chords ............................. 209
colored objects ....................................... 208
coloring notes ........................................ 208
coloring objects ...................................... 208, 585
coloring text .......................................... 695
coloring voices ....................................... 162
colors .................................................... 208
Colors, list of ....................................... 623

Appendix E: LilyPond index
fretted instruments, predefined string tunings........331
fretted instruments, right hand fingerings..............356
full-measure rests........................................55
Funk shape note heads.....................................37
funkHeads..................................................37
funkHeadsMinor............................................38

G
G clef.........................................................16
general-align..............................................228
ghost notes..................................................209
glissandi and repeats.....................................145
glissando.....................................................128
global variable.............................................704
glyph..........................................................702
glyphs, music...............................................103

G
G clef.........................................................16
general-align..............................................228
ghost notes..................................................209
glissandi and repeats.....................................145
glissando.....................................................128
global variable.............................................704
glyph..........................................................702
glyphs, music...............................................103
grow-direction.............................................89
grobs, overwriting.........................................585
grobs, visibility of.........................................584
groups..........................................................83

H
hairpin.........................................................114
hairpins, angled.............................................590
hairpins, constante.........................................117
hairpins, Ferneyhough.....................................117
hairpins, flared..............................................117
halfopen.......................................................699
halign.........................................................227
hammer on...................................................329
| Harmonic indications in tablature notation | 322 |
| Harmonic note heads | 34 |
| Harmonica Sacra note heads | 37 |
| harmonicByFret | 738 |
| harmonicByRatio | 738 |
| harmonicNote | 738 |
| harmonics on fretted instruments | 358 |
| harmonics, artificial | 314 |
| harmonics, natural | 313 |
| harmonicsOn | 738 |
| harp pedal diagrams | 311 |
| harp pedals | 311 |
| harps | 311 |
| hbracket | 230 |
| help, balloon | 211 |
| hidden notes | 207 |
| hide | 738 |
| hideKeySignature | 373 |
| hideNotes | 207 |
| hideStaffSwitch | 306 |
| hiding ancient staves | 190 |
| hiding of staves | 189 |
| hiding rhythmic staves | 190 |
| horizontal bracket | 214 |
| horizontal spacing | 530 |
| horizontal spacing, overriding | 602 |
| horizontal text alignment | 227 |
| horizontal-shift | 501 |
| Horizontal_bracket_engraver | 214 |
| horizontally centering text | 654 |
| hufnagel | 406, 406 |
| huge | 204, 226 |
| hymns | 289 |
| hyphens | 251 |

I

|ictus | 699 |
|images, embedding | 232 |
|immutable objects | 703 |
|immutable properties | 703 |
|implicit contexts | 563 |
|importing stencils into text, improvisation | 695 |
|improvisation | 40 |
|improvisationOff | 40, 74 |
|improvisationOn | 40, 74 |
|include-settings | 478 |
|including files | 472 |
|indent | 193, 501, 354 |
|indicating No Chord in ChordNames | 390 |
|indicating position and barring for fretted instruments | 357 |
|inlining an Encapsulated PostScript image | 672 |
|inner-margin | 500 |
|inserting music into text | 679 |
|inserting PostScript directly into text | 675 |
|inserting URL links into text | 676 |
|inStaffSegno | 738 |
|instrument names | 192, 486 |
|instrument names, adding to other contexts | 194 |
|instrument names, centering | 193 |
|instrument names, changing | 194 |
|instrument names, complex | 192 |
|instrument names, short | 192 |

J

|jazz chords | 392 |
|justified text | 229 |
|justified-lines | 235 |
|justify | 229 |
|justifying lines of text | 696 |
|justifying text | 661 |

K

|keep tagged music | 475 |
|keepWithTag | 739 |
|key | 20, 38, 739 |
|key signature | 5, 20, 413, 417 |
|key signature, visibility following explicit change | 587 |
|keyboard instrument staves | 302 |
|keyboard music, centering dynamics | 302 |
|keyed instrument staves | 302 |
|KievanStaffContext | 427 |
|KievanVoiceContext | 427 |
|killCues | 202, 739 |
|kirchenpausen | 57 |
|knee gap, with beams | 76 |

L

|label | 739 |
laissez vibrer | 50 |
laissezVibrer | 50 |
|language | 739 |
|language, note names in other | 7 |
|language, pitch names in other | 7 |
|languageRestore | 739 |
|languageSaveAndChange | 739 |
|large | 204, 226 |
|larger | 224, 226 |
|last-bottom-spacing | 499 |
|layers | 585 |
|layout file | 506 |
|layout interface | 564 |
|layout objects | 702 |
|ledger lines | 182 |
|ledger lines, internal | 182 |
|ledger lines, modifying | 182 |
|left-aligning text | 662 |
|left-align | 227 |
neomensural .............................................. 406
nested repeat ........................................... 145
nested staff brackets ................................. 179
nesting of staves ........................................ 179
new contexts ............................................. 547
new dynamic marks ...................................... 119
new spacing area ....................................... 532
new staff ............................................... 175
niente, al ................................................. 117
no chord symbol .......................................... 390
no reset accidental style ............................... 30
no-reset ................................................ 30
\noBeam .................................................. 86
non-ASCII characters ................................... 478
non-empty texts ......................................... 216
non-musical symbols ................................... 231
non-text fonts in markup ............................. 236
nonstaff-nonstaff-spacing ......................... 515
nonstaff-relatedstaff-spacing ..................... 515
nonstaff-unrelatedstaff-spacing .................. 515	noPageBreak ............................................ 739	noPageTurn ............................................. 739
normal repeat ........................................... 138
normal-size-super ...................................... 225
\normalsize .................................................. 204
notation font size ...................................... 204
notation inside markup ................................ 233
notation, explaining ................................... 211
notation, graphic ....................................... 231
note cluster .............................................. 158
note collisions .......................................... 162
note duration, default ................................. 42
note durations .......................................... 41
note grouping bracket ................................ 214
note head styles ........................................ 34, 645
note heads ................................................. 204
note heads, Aiken ...................................... 37
note heads, ancient .................................... 411, 429
note heads, Christian Harmony ..................... 37
note heads, cross ....................................... 34
note heads, diamond ................................... 34
note heads, diamond-shaped .......................... 313
note heads, easy notation ............................. 36
note heads, easy play ................................... 36
note heads, Funk ........................................ 37
note heads, guitar ...................................... 34
note heads, harmonic ................................... 34
note heads, Harmonica Sacra ....................... 37
note heads, improvisation ............................ 40
note heads, lute ......................................... 34
note heads, practice .................................... 36
note heads, sacred harp ................................ 37
note heads, shape ....................................... 37
note heads, slashed .................................... 40
note heads, Southern Harmony ..................... 37
note heads, special ..................................... 34
note lengths ............................................. 41
note names, default .................................... 5
note names, Dutch ...................................... 5
note names, other languages ........................ 7
note-event ............................................... 197
\Note_heads_engraver .................................. 72
notes within text by log and dot-count ................ 678
notes within text by string ............................. 678
notes, colored ........................................... 208
notes, colored in chords .............................. 209
notes, cross-staff ...................................... 303, 307
notes, dotted ........................................... 42
notes, double-dotted ................................... 42
notes, ghost ............................................. 209
notes, hidden ........................................... 207
notes, invisible ......................................... 207
notes, parenthesized ................................... 209
notes, smaller .......................................... 198
notes, spacing horizontally ......................... 532
notes, splitting .......................................... 72
notes, transparent ...................................... 207
notes, transposition of ............................... 10
null ........................................................ 227
nullVoice .................................................. 288
numbers, bar ............................................. 96
numbers, measure ...................................... 96
\numericTimeSignature .................................. 60

O
objects, colored ........................................... 208
objects, coloring ....................................... 585
objects, overwriting ................................... 585
objects, rotating ........................................ 590
objects, visibility of .................................... 584
octavation ................................................ 22
octave changing mark ................................... 1
octave check .............................................. 9
octave correction ........................................ 9
octave entry, absolute .................................. 1
octave entry, relative ................................... 2
octave specification, absolute ....................... 1
octave specification, relative ....................... 2
octave transposition .................................... 17
octave transposition, optional ...................... 17
\octaveCheck ............................................. 9, 739
offset .................................................... 739
omit ....................................................... 740
on-the-fly ................................................ 459
once ....................................................... 740
\oneVoice ................................................ 159
open ....................................................... 111, 699
open string indication ................................. 313
operation, inversion .................................... 13
operation, modal inversion ........................... 15
operation, retrograde ................................... 13
\\operation, transposition ............................. 14
\\operations, modal ...................................... 14
optional octave transposition ....................... 17
oratorio ................................................... 278
orchestral strings ....................................... 312
organ pedal marks ...................................... 111
ornamentation .......................................... 111
ornaments ............................................... 104, 111
ossia ...................................................... 185, 190
ottoman ................................................... 22, 740
judge ....................................................... 440
\outer-margin ............................................ 500
output definitions ...................................... 545
output-count ............................................. 704
output-def ............................................... 703
output-suffix .......................................... 704
pitches .............................................. 1
Pitches in MIDI ................................. 486
pitches, transposition of ....................... 10
pitchnames ...................................... 704
pizzicato, Bartók ................................. 314
pizzicato, snap ................................... 314
placeholder events .............................. 155
placement of lyrics .............................. 253
placing horizontal brackets around text ... 672
placing parentheses around text ............ 673
placing vertical brackets around text ..... 669
pointAndClickOff ................................ 741
pointAndClickOn ................................ 741
pointAndClickTypes ............................. 741
polymetric, meters, with beams .............. 70
polymetric scores ................................ 554
polymetric signatures ........................... 69
polyphonic music ................................ 162
polyphony, shared lyrics ....................... 268
polyphony, single-staff ......................... 159
portato ........................................... 112, 699
positioning multi-measure rests ............ 57
postscript ....................................... 232
power chords .................................... 359
powerChords ..................................... 359
PP .................................................. 114
PPP .................................................. 114
PPPP ............................................... 114
practice note heads ............................. 36
prall ............................................... 111, 699
prall, down ........................................ 111
prall, up .......................................... 111
pralldown ......................................... 699
prallmordent ..................................... 111, 699
prallprall ......................................... 111, 699
prallup ............................................ 699
predefined string tunings for fretted instruments ........................................ 331
predefinedFretboardsOff ....................... 353
predefinedFretboardsOn ....................... 353
prima volta ....................................... 138
print-all-headers ................................ 503
print-first-page-number ......................... 503
print-page-number ................................ 503
printing chord names ........................... 389
printing order .................................... 585
printing reserved characters ................. 223
printing special characters ................. 223
prob ............................................... 704
properties ........................................ 567
properties, grob .................................. 569
property object .................................. 704
psalms ............................................ 289
pull off ........................................... 329
punctuation in lyrics ........................... 240
pure containers, Scheme ....................... 602
pushToTag ......................................... 741
putting space around text ..................... 663

Q
quarter tones .................................... 5
Quarter tones in MIDI .......................... 486
quarter-tone accidental ........................ 7
quote, voices ..................................... 95
quoted text ....................................... 216
quoted text in markup mode .................. 223
quotedCueEventTypes ......................... 197
quotedEventTypes ............................... 197
quoteDuring ..................................... 195, 198, 741
quotes in lyrics .................................. 240
quotes, in lyrics .................................. 247

R
r ..................................................... 52
R ...................................................... 55
ragged-bottom ................................. 497
ragged-last ....................................... 500, 534
ragged-last-bottom ............................. 497
ragged-right ...................................... 500, 534
railroad tracks .................................... 127
raise ............................................... 227
raising text ....................................... 664
rallentando in MIDI ............................. 486
range of pitches ................................. 32
Ratisbona, Editio ................................ 406
referencing contexts ........................... 547
referencing page labels in text .............. 696
referencing page numbers in text ........... 693
regular line breaks ............................. 508
rehearsal mark format ......................... 102
rehearsal mark style ............................ 102
rehearsal mark, manual ......................... 102
rehearsal marks .................................. 101
relative ........................................... 2, 5, 12, 305, 741
relative octave entry ........................... 2
relative octave entry and chords ............ 4
relative octave entry and transposition .... 5
relative octave specification ................. 2
relative pitch, chords ......................... 155
religious music .................................. 289
reminder accidental ............................ 6
removals, in chords ............................. 388
remove tagged music ............................ 475
RemoveEmptyStaves ............................. 744
removeWithTag .................................. 741
removing cue notes ............................. 202
renaissance music ............................... 178
repeat and measure number .................. 145
repeat bars ....................................... 90
repeat number, changing ...................... 145
repeat timing information ..................... 145
repeat volta, changing ......................... 145
repeat with alternate endings ............... 138
repeat with anacrusis ......................... 140
repeat with pickup ............................. 140
repeat with upbeat ............................. 140
repeat, ambiguous .............................. 145
repeat, end ....................................... 145
repeat, manual ................................... 145
repeat, measure .................................. 150
repeat, nested ................................... 145
repeat, normal ................................... 138
repeat, percent ................................... 150
repeat, short .................................... 150
repeat, start ........................................ 145
repeat, tremolo .................................... 152
repeatCommands .................................. 145
repeating lyrics with alternative endings .... 263
repeating ties ................................... 50
repeats ............................................ 92
repeats and glissandi .............................. 145
repeats and lyrics ................................ 259
repeats and slur ................................... 145
repeats in MIDI ................................... 487
repeats, alternative ............................... 148
repeats, alternative bar numbers ................. 144
repeats, bar numbers letters ...................... 144
repeats, unfold .................................. 148
repeats, with segno ................................ 141
repeats, with ties ................................ 141
repeats, written-out ................................ 148
repeatTie .......................................... 50
repetition, using q ................................ 156, 320
reserved characters, printing ..................... 223
resetRelativeOctave ................................ 741
resetting of staves ................................ 185
rest ................................................. 52
rest, church ....................................... 57
rest, collisions of ................................ 59
rest, condensing ordinary ......................... 59
rest, entering durations ......................... 52
rest, full-measure ................................ 55
rest, invisible .................................... 54
rest, multi-measure ................................ 53, 55
rest, specifying vertical position ................. 53
rest, whole for a full measure ................. 55
rest, whole-measure ............................... 53
rest-event ......................................... 197
restGroup ......................................... 185
resting default properties for time signatures .. 61
restrainOpenStrings ................................ 319
rests or multi-measure rests within text by string
   dot-count ..................................... 678
   rests ........................................... 678
   rests, ancient .................................. 412
   rests, splitting ................................ 72
   retrograde .................................... 13, 741
   retrograde transformation .................... 13
   reverseReason ................................ 111, 699
reverting overrides ................................ 570
revertTimeSignatureSettings ..................... 741
rfz .................................................. 114
rgb color ......................................... 209
rgb-color .......................................... 209
rhe ............☆................................... 699
rhythmic staff .................................... 175
RhythmicStaff .................................... 175
rhythms in MIDI .................................. 486
rhythms, showing melody ......................... 73
right aligning text ................................ 665
right hand fingerings for fretted instruments .. 356
right-align ........................................ 227
right-margin ..................................... 500
rightHandFinger .................................. 356, 742
root of chord ..................................... 385
rotating objects .................................. 590
rotating text ..................................... 665
rounded-box ....................................... 220
rtoe .................................................. 699
S
s ..................................................... 54
sacred harp note heads ......................... 37
sacredHarpHeads .................................. 37
sacredHarpHeadsMinor ......................... 38
SATB .................................................. 278
scalable vector graphics output ................. 482
scaleDurations ................................... 48, 69, 742
scaling durations ................................ 48
scaling markup ................................... 676
scaling text ....................................... 666
Scheme object ..................................... 704
Scheme variable .................................. 704
Scheme, pure containers ......................... 602
Scheme, unpure containers ...................... 602
score inside markup ................................ 234
score-markup-spacing ............................. 498
score-system-spacing ............................. 499
scoreTitleMarkup ................................ 456
Scottish highland bagpipe ....................... 373
script on multi-measure rest .................... 56
scripts ............................................. 111
seconda volta .................................... 138
segno .............................................. 92, 103, 111, 699
segno on bar line ................................ 219
segno, with repeats ................................ 141
selecting font size (notation) ................. 204
self-alignment-X .................................. 515
Semai form ....................................... 438
semi-flat .......................................... 8
Semi-flat symbol appearance ...................... 436
semi-flats ......................................... 5
semi-sharp ....................................... 8
semi-sharps ....................................... 5
semicirculus ...................................... 699
separate text .................................... 221
sesqui-flat ....................................... 8
sesqui-sharp ..................................... 8
set ................................................... 78
set-octavation ................................... 22
setting extent of text objects .................. 696
setting horizontal text alignment .............. 658
setting subscript in standard font size ........ 649
setting superscript in standard font size ....... 649
settingsFrom ..................................... 742
seventh chords .................................... 385
sf .................................................... 114
sff .................................................. 114
sfs .................................................. 114
shape ............................................... 742
shape notes ....................................... 37
shaping slurs and ties ............................ 598
shared properties ................................ 703
sharp .............................................. 5
sharp, double .................................... 5
shift note ......................................... 162
shift, rest, automatic ............................ 162
shiftDurations .................................... 742
shifting voices .................................... 162
shiftLeft .......................................... 162
shiftOn ........................................... 162
shiftOff .......................................... 162
shiftOnn .......................................... 162
shiftOnnn ........................................ 162
short-indent .................................... 193, 501
Appendix E: LilyPond index

- text in columns .............................................. 228
- text in volta bracket .................................. 224
- text items, non-empty .................................. 147
- text marks .................................................. 219
- text markup ............................................... 223
- text on bar line ......................................... 219
- text on multi-measure rest ............................. 56
- text outside margin ...................................... 217
- text padding .............................................. 231
- Text scripts ............................................... 216
- text size .................................................... 224
- text spanners ............................................. 217
- text spanners, formatting ............................... 217
- text spread over multiple pages ..................... 235
- text, aligning ............................................. 227
- text, centering on the page ............................ 229
- text, decorating .......................................... 230
- text, framing .............................................. 230
- text, horizontal alignment .............................. 227
- text, justified ............................................ 229
- text, keeping inside margin ............................ 217
- text, multi-line ........................................... 228
- Text, other languages .................................... 216
- text, separate ........................................... 221
- text, standalone .......................................... 221
- text, top-level ............................................ 221
- text, vertical alignment ................................. 226
- text, word-wrapped ...................................... 229
- textLengthOff ............................................. 57
- textLengthOff ............................................. 57
- textLengthOn ............................................. 57
- textLengthOn ............................................. 57
- textSpannerDown .......................................... 217
- textSpannerNeutral ....................................... 217
- textSpannerUp ............................................ 217
- Thorough bass ............................................ 398
- thumb ...................................................... 205
- thumb ...................................................... 205
- thumb marking ........................................... 699
- thumb-script ............................................. 205
- tick mark ................................................... 126
- tie .......................................................... 49
- tied note, accidental ................................... 6
- tieDashed ................................................... 50
- tieDashPattern ........................................... 743
- tieDotted ................................................... 50
- tieDown ..................................................... 50
- tieNeutral .................................................. 50
- ties and chords .......................................... 49
- ties and volta brackets ................................. 50
- ties, alternative endings ................................ 141
- ties, appearance ......................................... 50
- ties, dashed .............................................. 50
- ties, dotted ............................................... 50
- ties, in lyrics ............................................ 247
- ties, in repeats ......................................... 141
- ties, laissez viber ........................................ 50
- ties, modifying .......................................... 597
- ties, placement .......................................... 50
- ties, repeating .......................................... 50
- tieSolid ...................................................... 50
- tieUp ........................................................ 50
- time ........................................................ 59, 78, 743
- time administration ..................................... 110
- time signature ........................................... 59
- time signature default settings ....................... 60
- time signature properties, restoring default values ........................................ 61
- time signature style ..................................... 60
- time signature, compound ............................. 71
- time signature, visibility of ........................... 60
- time signatures ........................................... 410
- time signatures, double ................................. 69
- Time signatures, multiple ................................ 554
- time signatures, polymeric ............................. 69
- times ........................................................ 743
- timeSignatureFraction .................................. 69
- timing (within the score) .............................. 110
- timing information and repeats ....................... 145
- tiny ........................................................ 204, 226
- tocItem ...................................................... 743
- Top .......................................................... 1
- top-level text ............................................. 221
- top-margin ................................................ 497
- top-markup-spacing ...................................... 499
- top-system-spacing ...................................... 499
- toplevel-bookparts ...................................... 704
- toplevel-scores .......................................... 704
- transcription of mensural music ....................... 178
- transformation, retrograde ............................ 13
- transformations, modal ................................ 14
- translate .................................................... 228
- translate-scaled .......................................... 228
- translating text .......................................... 666
- transparent notes ........................................ 207
- transparent, making objects ........................... 584
- transpose .................................................. 5, 10, 12, 743
- transposed clefs, visibility of ........................ 589
- transposedCueDuring ..................................... 201, 743
- transposing ................................................ 10
- transposing clef .......................................... 17
- transposing fret diagrams ............................. 345
- transposing instrument ................................ 24
- transposing instruments ................................ 10
- transposition ............................................. 10
- transposition ............................................. 24, 195, 743
- transposition and relative octave entry .............. 5
- transposition of notes .................................. 10
- transposition of pitches ............................... 10
- transposition, instrument .............................. 24
- transposition, MIDI ...................................... 24
- transposition, modal .................................... 14
- tre corde ................................................... 309
- treble clef ................................................ 16
- treCorde ................................................... 309
- tremolo ..................................................... 152
- tremolo beams ............................................ 152
- tremolo marks ............................................ 152
- tremolo, cross-staff .................................... 153
- tremoloFlags ............................................. 152
- triads ....................................................... 385
- triangle ..................................................... 231
- trill ......................................................... 111
- trill ......................................................... 135
- trill ......................................................... 699
- trill with accidental ..................................... 137
- trills ......................................................... 135
- trills in MIDI ............................................... 486
- trills, pitched ............................................ 136
- triplet formatting ........................................ 44
Appendix E: LilyPond index

triplets ........................................... 44
tuning, non-Western ............................. 434
tunings, banjo .................................. 361
tuplet ............................................ 44, 69, 743
tuplet bracket placement ........................ 44
tuplet formatting .................................. 44
tuplet grouping ..................................... 44
Tuplet number changes ........................... 45
tupletDown ......................................... 44
tupletNeutral ....................................... 44
TupletNumber ....................................... 45
tupletNumberFormatFunction .................... 44
tupletSpan .......................................... 743
tupletSpannerDuration ............................ 44
tupletUp ........................................... 44
Turkish music ....................................... 440
Turkish note names ................................ 440
turn ................................................... 111, 699
turns in MIDI ....................................... 486
tweak .................................................. 743
tweak, relation to \override ....................... 573
tweaking ............................................ 571
tweaking control points ........................... 573
tweaking grace notes .............................. 105
two-sided ........................................... 500
type ................................................... 560
typeface ............................................. 702
typeset text ........................................ 223

U

U.C. .................................................. 309
ukulele .............................................. 335
una corda ........................................... 309
unaCorda ............................................ 309
underline .......................................... 224
underlining text ................................... 653
undo .................................................. 744
unfold ............................................... 148
unfold repeat ....................................... 148
unfold repeat, alternate endings ............... 148
unfoldRepeats ..................................... 744
unHideNotes ....................................... 207
Unicode ............................................ 479
unmetered music .................................. 68, 110
unmetered music, accidentals .................... 68
unmetered music, bar lines ....................... 68
unmetered music, bar numbers .................... 68
unmetered music, beams ........................... 68
unmetered music, line breaks ..................... 69
unmetered music, page breaks .................... 69
unpure containers, Scheme ....................... 602
up bow indication .................................. 313
upbeat ............................................... 67
upbeat in a repeat .................................. 140
upbow .............................................. 111, 699
upmordent ......................................... 699
upprrall ............................................ 699
UTF-8 ............................................... 478

V

varbaritone clef ................................... 16
varcoda ............................................. 111, 699
variables ........................................... 447
variables, use of .................................. 474
Vaticana, Editio .................................... 405, 406
VaticanaStaff ....................................... 175
VaticanaStaffContext .............................. 416
VaticanaVoiceContext .............................. 416
vertical lines between staves .................... 212
vertical positioning of dynamics ............... 116
vertical spacing ................................... 515, 535
vertical text alignment ........................... 227
VerticalAxisGroup .................................. 515
vertically centering text .......................... 666
verylongfermata .................................. 699
violin clef ......................................... 16
visibility of objects ................................ 584
visibility of transposed clefs .................... 589
voice ................................................. 25, 27
voice ................................................. 159
Voice ................................................ 159
voice accidental style ............................. 27
voice styles ........................................ 162
voice, following ................................... 306
voiceOne .......................................... 159
voices, \partcombine with \autoBeamOff .......... 77
voices, divided .................................... 280
voices, multiple ................................... 162
voices, quoting .................................... 195, 198
void ................................................... 744
volta ................................................. 138
volta bracket ....................................... 145
volta bracket with text ......................... 147
volta brackets and ties ......................... 138
volta, prima ....................................... 138
volta, seconda ..................................... 138

W

Walker shape note heads .......................... 37
walkerHeads ....................................... 37
walkerHeadsMinor .................................. 38
whichBar ............................................ 96
White mensural ligatures ......................... 414
whitespace ......................................... 447
whole rest for a full measure .................... 55
wind instruments ................................... 370
with-color .......................................... 208
withMusicProperty ................................ 744
wordwrap .......................................... 229
wordwrap-lines ................................... 235
wordwrapped text ................................. 229
writing music in parallel ....................... 172
written-out repeats ............................... 148

X

X-offset ............................................ 515
x11 color ........................................... 208, 209
x11-color .......................................... 208, 209
xNote .............................................. 744
xNotesOn .......................................... 744