

# LilyPond

---

The music typesetter

## Snippets

### The LilyPond development team

This document shows a selected set of LilyPond snippets from the LilyPond Wiki (<https://wiki.lilypond.community>). It is in the public domain.

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 <https://lilypond.org/>.

This document has been placed in the public domain.

For LilyPond version 2.25.32

---

# Table of Contents

<b>Preface</b> .....	<b>1</b>
----------------------	----------

## Musical notation

<b>1 Pitches</b> .....	<b>5</b>
Adding ambitus per voice .....	5
Adding an ottava marking to a single voice .....	5
Aiken head thin variant noteheads .....	6
Altering the length of beamed stems .....	6
Ambitus .....	7
Ambitus after key signature .....	7
Ambitus with multiple voices .....	8
Applying note head styles depending on the step of the scale .....	8
Automatically changing the stem direction of the middle note based on the melody .....	9
Changing ottava text .....	10
Changing the ambitus gap .....	10
Changing the interval of lines on the staff .....	11
Clefs can be transposed by arbitrary amounts .....	12
Coloring notes depending on their pitch .....	12
Creating a sequence of notes on various pitches .....	13
Creating custom key signatures .....	13
Direction of merged ‘fa’ shape note heads .....	14
Force a cancellation natural before accidentals .....	14
Forcing a clef symbol to be displayed .....	15
Generating random notes .....	15
Hiding accidentals on tied notes at the start of a new system .....	16
Keep change clefs full-sized .....	16
Makam example .....	16
Modifying the ottava spanner slope .....	17
Non-traditional key signatures .....	17
Numbers as easy note heads .....	18
Orchestra, choir and piano template .....	19
Preventing extra naturals from being automatically added .....	24
Preventing natural signs from being printed when the key signature changes .....	24
Quoting another voice with transposition .....	24
Separating key cancellations from key signature changes .....	25
Transposing pitches with minimum accidentals (“smart” transpose) .....	26
Turkish Makam example .....	28
Tweaking clef properties .....	29
Using \autoChange with more than one voice .....	30
<b>2 Rhythms</b> .....	<b>32</b>
Adding beams, slurs, ties, etc., when using tuplet and non-tuplet rhythms .....	32
Adding drum parts .....	32
Adjusting grace note spacing .....	33
Aligning bar numbers .....	33

Alternative breve notes .....	34
Appoggiatura or grace note before a bar line .....	35
Automatic beam subdivisions .....	35
Automatically change durations .....	36
Beam endings in Score context .....	37
Beam nibs .....	38
Beams across line breaks .....	39
Changing beam knee gap .....	40
Changing form of multi-measure rests .....	40
Changing the number of augmentation dots per note .....	40
Changing the tempo without a metronome mark .....	41
Changing the tuplet number .....	41
Changing time signatures inside a polymetric section using <code>\scaleDurations</code> .....	41
Chant or psalm notation .....	43
Compound time signatures .....	43
Conducting signs, measure grouping signs .....	44
Controlling tuplet bracket visibility .....	44
Cow and ride bell example .....	45
Creating metronome marks in markup mode .....	46
Engraving ties manually .....	46
Engraving tremolos with floating beams .....	47
Entering several tuplets using only one <code>\tuplet</code> command .....	48
Forcing rehearsal marks to start from a given letter or number .....	48
Generating custom flags .....	49
Guitar strum rhythms .....	50
Heavily customized polymetric time signatures .....	51
High and low woodblock example .....	52
Making an object invisible using the ‘transparent’ property .....	52
Making slurs with complex dash structure .....	53
Manually controlling beam positions .....	54
Merging multi-measure rests in a polyphonic part .....	54
Modifying tuplet bracket length .....	54
Moving dotted notes in polyphony .....	55
Multi-measure rest length control .....	55
Multi-measure rest markup .....	56
Non-default tuplet numbers .....	56
Numbering single measure rests .....	57
Partcombine and <code>\autoBeamOff</code> .....	58
Percussion example .....	58
Permitting line breaks within beamed tuplets .....	59
Positioning grace note beams at the height of normal note beams .....	60
Positioning grace notes with floating space .....	61
Positioning multi-measure rests .....	62
Positioning opposing fermatas on a bar line .....	63
Preventing final mark from removing final tuplet .....	63
Printing bar numbers at regular intervals .....	64
Printing bar numbers for broken measures .....	64
Printing bar numbers inside boxes or circles .....	65
Printing bar numbers using <code>modulo-bar-number-visible</code> .....	66
Printing bar numbers with changing regular intervals .....	66
Printing metronome and rehearsal marks below the staff .....	67
Printing music with different time signatures .....	67

Printing the bar number for the first measure .....	69
Printing tuplet brackets on the note head side .....	70
Redefining grace note global defaults .....	70
Removing bar numbers from a score .....	71
Rest styles .....	71
Reverting default beam endings .....	72
Rhythmic slashes .....	73
Skips in lyric mode .....	73
Skips in lyric mode (2) .....	74
Stemlets .....	74
Strict beat beaming .....	74
Subdividing beams .....	75
Tam-tam example .....	76
Tambourine example .....	77
Three-sided box .....	77
Time signature in brackets .....	78
Time signature in parentheses .....	78
Time signature printing only the numerator as a number (instead of the fraction) .....	78
Tweaking grace layout within music .....	79
User-defined time signatures .....	79
Using alternative flag styles .....	80
Using grace note slashes with normal heads .....	81
Using ties with arpeggios .....	82
 <b>3 Expressive marks .....</b>	 <b>83</b>
Adding parentheses around an expressive mark or chordal note .....	83
Adding timing marks to long glissandi .....	83
Adjusting slur positions vertically .....	84
Adjusting the shape of falls and doits .....	85
Aligning the ends of hairpins to NoteColumn directions .....	85
Alternative breve notes .....	85
Asymmetric slurs .....	86
Breathing signs .....	86
Broken crescendo hairpin .....	87
Caesura (“railtracks”) with fermata .....	87
Center text below hairpin dynamics .....	88
Changing text and spanner styles for text dynamics .....	89
Changing the appearance of a slur from solid to dotted or dashed .....	90
Changing the breath mark symbol .....	90
Changing the number of augmentation dots per note .....	91
Combining dynamics with markup texts .....	91
Combining dynamics with markup texts (2) .....	91
Contemporary glissando .....	92
Controlling spanner visibility after a line break .....	92
Controlling the placement of chord fingerings .....	93
Controlling the vertical ordering of scripts .....	94
Creating “real” parenthesized dynamics .....	94
Creating a delayed turn .....	95
Creating arpeggios across notes in different voices .....	95
Creating cross-staff arpeggios in a piano staff .....	96
Creating cross-staff arpeggios in other contexts .....	96

Creating double-digit fingerings .....	97
Creating slurs across voices .....	97
Creating text spanners .....	98
Dynamics spanner with custom text .....	99
Glissandi can skip grobs .....	99
Hairpins with different line styles .....	100
Hiding the extender line for text dynamics .....	100
Horizontally aligning custom dynamics like “più f” .....	100
Inserting a caesura .....	103
Laissez vibrer ties .....	104
Line arrows .....	104
Making slurs with complex dash structure .....	105
Modifying default values for articulation shorthand notation .....	105
Moving slur positions vertically .....	106
Moving the ends of hairpins .....	106
Positioning arpeggios .....	107
Positioning text markups inside slurs .....	107
Printing hairpins in various styles .....	108
Printing hairpins using al niente notation .....	108
Printing metronome and rehearsal marks below the staff .....	109
Setting hairpin behavior at bar lines .....	109
Setting the minimum length of hairpins .....	109
Showing the same articulation above and below a note or chord .....	110
Snap pizzicato (“Bartok” pizzicato) .....	110
Using \arpeggioBracket to make divisi more visible .....	111
Using a tick as the breath mark symbol .....	111
Using double slurs for legato chords .....	112
Using the whiteout property .....	112
Vertical line as a baroque articulation mark .....	113
Vertically aligning dynamics across multiple notes .....	113
 <b>4 Repeats .....</b>	 <b>114</b>
Changing the default bar lines .....	114
Controlling the appearance of tremolo slashes .....	114
Cross-staff tremolos .....	116
Engraving tremolos with floating beams .....	116
Isolated percent repeats .....	117
Measure counters .....	117
Percent repeat count visibility .....	118
Percent repeat counter .....	119
Positioning segno and coda (with line break) .....	119
Setting the double repeat default for volte .....	120
Shortening volta brackets .....	120
Unfolding tremolo repeats .....	121
Volta below chords .....	122
Volta brackets in multiple staves .....	122
Volta text markup using repeatCommands .....	123
 <b>5 Simultaneous notes .....</b>	 <b>124</b>
Additional voices to avoid collisions .....	124
Changing \partCombine texts .....	124

Changing a single note's size in a chord .....	125
Clusters .....	125
Combining two parts on the same staff .....	126
Displaying complex chords .....	127
Forcing horizontal shift of notes .....	127
Making an object invisible using the 'transparent' property .....	128
Moving dotted notes in polyphony .....	129
Suppressing warnings for clashing note columns .....	129
Two \partCombine pairs on one staff .....	129
<b>6 Staff notation .....</b>	<b>132</b>
Adding ambitus per voice .....	132
Adding an extra staff .....	132
Adding an extra staff at a line break .....	133
Adding indicators to staves which get split after a break .....	134
Adding orchestral cues to a vocal score .....	138
Alternative bar numbering .....	140
Ambitus after key signature .....	141
Changing the default bar lines .....	142
Changing the number of lines in a staff .....	142
Changing the staff size .....	143
Creating blank staves .....	143
Creating custom key signatures .....	145
Cross-staff stems .....	146
Display bracket with only one staff in a system .....	147
Displaying a whole GrandStaff system if only one of its staves is alive .....	147
Extending a trill spanner .....	149
Extending glissandi across repeats .....	150
Flat ties .....	151
Forcing measure width to adapt to a metronome mark's width .....	153
Glissandi can skip grobs .....	154
Incipit .....	154
Inserting score fragments above a staff, as markups .....	159
Let TabStaff print the topmost string at bottom .....	160
Letter tablature formatting .....	161
Making glissandi breakable .....	161
Making some staff lines thicker than the others .....	163
Measure counters .....	163
Mensurstriche layout (bar lines between the staves) .....	164
Modifying the ottava spanner slope .....	164
Nesting staves .....	165
Non-traditional key signatures .....	166
Orchestra, choir and piano template .....	167
Print chord names with same root and different bass as slash and bass note .....	171
Putting lyrics inside the staff .....	173
Quoting another voice .....	173
Quoting another voice with transposition .....	174
Reconciliating bar lines thickness when staves are of different sizes (TODO duplicates snippet in Documentation/en/notation/spacing.itely) .....	175
Removing brace on first line of piano score .....	177
Removing connecting bar lines on StaffGroup, PianoStaff, or GrandStaff .....	178

Removing the first empty line .....	179
Setting system separators .....	180
Shape individual ties in chords .....	181
Tick bar lines .....	184
Time signature in brackets .....	184
Time signature in parentheses .....	184
Tweaking clef properties .....	185
Two \partCombine pairs on one staff .....	186
Use square bracket at the start of a staff group .....	188
Using \autoChange with more than one voice .....	188
Using mark lines in a Frenched score .....	189
Vertically aligned StaffGroups without connecting SystemStartBar .....	192
Volta below chords .....	198
Volta brackets in multiple staves .....	199
 <b>7 Editorial annotations .....</b>	 <b>200</b>
Adding fingerings to a score .....	200
Adding links to objects .....	200
Adding markups in a tablature .....	202
Allowing fingerings to be printed inside the staff .....	202
Alternative bar numbering .....	203
Analysis brackets above the staff .....	204
Analysis brackets with labels .....	205
Applying note head styles depending on the step of the scale .....	205
Blanking staff lines using the \whiteout command .....	206
Changing a single note's size in a chord .....	207
Changing the appearance of a slur from solid to dotted or dashed .....	207
Coloring notes depending on their pitch .....	207
Controlling the placement of chord fingerings .....	208
Creating a delayed turn .....	209
Creating blank staves .....	210
Creating double-digit fingerings .....	211
Default direction of stems on the center line of the staff .....	212
Different font size settings for instrumentName and shortInstrumentName .....	212
Drawing boxes around grobs .....	213
Drawing circles around note heads .....	214
Drawing circles around various objects .....	214
Embedding native PostScript in a \markup block .....	215
Generate special note head shapes .....	215
Grid lines: changing their appearance .....	216
Grid lines: emphasizing rhythms and notes synchronization .....	217
Hammer-on and pull-off .....	218
Hammer-on and pull-off using chords .....	218
Hammer-on and pull-off using voices .....	219
Making some staff lines thicker than the others .....	219
Marking notes of spoken parts with a cross on the stem (Sprechstimme) .....	219
Measure counters .....	220
Measure spanner .....	221
Positioning fingering indications precisely .....	222
Positioning text markups inside slurs .....	223
Printing text from right to left .....	223

String number extender lines .....	224
Using the whiteout property .....	224
<b>8 Text .....</b>	<b>225</b>
Adding markups in a tablature .....	225
Adding the current date to a score .....	225
Adjusting vertical spacing of lyrics .....	226
Aligning and centering instrument names .....	226
Aligning syllables with melisma .....	228
Aligning text marks to notes .....	228
Blanking staff lines using the \whiteout command .....	229
Center text below hairpin dynamics .....	229
Changing ottava text .....	231
Changing the default text font family .....	231
Combining dynamics with markup texts .....	232
Combining dynamics with markup texts (2) .....	233
Combining two parts on the same staff .....	233
Creating “real” parenthesized dynamics .....	234
Creating text spanners .....	235
Demonstrating all \header fields .....	236
Embedding native PostScript in a \markup block .....	237
Formatting lyrics syllables .....	237
How to put ties between syllables in lyrics .....	238
Lyrics alignment .....	238
Markup list .....	239
Multi-measure rest markup .....	240
Of the ubiquity of markup objects .....	240
Outputting the version number .....	242
Piano template with centered lyrics .....	242
Printing marks on every staff .....	243
Printing text from right to left .....	243
Putting lyrics inside the staff .....	244
Stand-alone two-column markup .....	244
String number extender lines .....	245
Three-sided box .....	245
UTF-8 .....	246
Vocal ensemble template with lyrics aligned below and above the staves .....	247
Volta text markup using repeatCommands .....	249

## Specialist notation

<b>9 Vocal music .....</b>	<b>253</b>
Adding ambitus per voice .....	253
Adding indicators to staves which get split after a break .....	253
Adding orchestral cues to a vocal score .....	257
Adjusting vertical spacing of lyrics .....	259
Aligning syllables with melisma .....	259
Ambitus .....	260
Ambitus after key signature .....	261



Ambitus with multiple voices .....	261
Ancient notation template – modern transcription of Gregorian music .....	262
Anglican psalm template .....	262
Arranging separate lyrics on a single line .....	265
Changing stanza fonts .....	266
Chant or psalm notation .....	267
Forcing hyphens to be shown .....	267
Formatting lyrics syllables .....	268
How to put ties between syllables in lyrics .....	268
Hymn template .....	268
Lyrics alignment .....	270
Marking notes of spoken parts with a cross on the stem (Sprechstimme) .....	271
Orchestra, choir and piano template .....	272
Piano template with melody and lyrics .....	277
Putting lyrics inside the staff .....	278
SATB choir template – four staves .....	278
Single-staff template with notes, lyrics, and chords .....	280
Single-staff template with notes, lyrics, chords, and frets .....	281
Single-staff template with notes and lyrics .....	282
Skips in lyric mode .....	282
Skips in lyric mode (2) .....	283
Using \arpeggioBracket to make divisi more visible .....	283
Using tags to produce mensural and modern music from the same source .....	284
Vertically aligning ossias and lyrics .....	286
Vertically aligning stanza numbers of different staves .....	287
Vertically centered common lyrics .....	288
Vocal ensemble template .....	289
Vocal ensemble template with automatic piano reduction .....	291
Vocal ensemble template with lyrics aligned below and above the staves .....	293
Vocal ensemble template with verse and refrain .....	295
 <b>10 Keyboard and other multi-staff instruments .....</b>	 <b>298</b>
Accordion register symbols .....	298
Changing the text for sustain markings .....	299
Clusters .....	299
Controlling the placement of chord fingerings .....	299
Creating slurs across voices .....	300
Cross-staff chords – beaming problems workaround .....	301
Cross-staff tremolos .....	302
Fine-tuning pedal brackets .....	302
Indicating cross-staff chords with arpeggio bracket .....	303
Jazz combo template .....	303
Laissez vibrer ties .....	309
Piano template (simple) .....	310
Piano template with centered lyrics .....	310
Piano template with melody and lyrics .....	311
Removing brace on first line of piano score .....	312
Using \autoChange with more than one voice .....	313
Vocal ensemble template with automatic piano reduction .....	314

<b>11</b>	<b>Unfretted string instruments</b>	<b>317</b>
	Creating slurs across voices	317
	Dotted harmonics	317
	Snap pizzicato (“Bartok” pizzicato)	318
	String quartet template (simple)	318
	String quartet template with separate parts	319
<b>12</b>	<b>Fretted string instruments</b>	<b>323</b>
	Adding fingerings to a score	323
	Adding fingerings to tablatures	323
	Adding markups in a tablature	323
	Allowing fingerings to be printed inside the staff	324
	Automatic fretboards barré	325
	Changing fret orientations	325
	Chord changes for fretboards	326
	Chord glissando in tablature	326
	Chords with stretched fingering for FretBoards and TabVoice	327
	Controlling the placement of chord fingerings	328
	Customizing fretboard fret diagrams	329
	Customizing markup fret diagrams	330
	Fingerings, string indications, and right-hand fingerings	332
	Flamenco notation	332
	Fret diagrams explained and developed	335
	Fretboards alternate tables	343
	Fretted-string harmonics in tablature	344
	Guitar slides	346
	Guitar strum rhythms	347
	Hammer-on and pull-off	348
	Hammer-on and pull-off using chords	348
	Hammer-on and pull-off using voices	348
	How to change fret diagram position	349
	Jazz combo template	350
	Laissez vibrer ties	355
	Let TabStaff print the topmost string at bottom	356
	Letter tablature formatting	357
	Open-string harmonics in tablature	357
	Placement of right-hand fingerings	359
	Polyphony in tablature	360
	Setting up predefined fretboards for other instruments	361
	Slides in tablature	363
	Stem and beam behavior in tablature	364
	String number extender lines	364
<b>13</b>	<b>Percussion</b>	<b>366</b>
	Adding drum parts	366
	Cow and ride bell example	367
	Heavily customized polymetric time signatures	367
	High and low woodblock example	368
	Jazz combo template	369
	Percussion beaters	374

Percussion example .....	377
Printing music with different time signatures .....	379
Tam-tam example .....	381
Tambourine example .....	381
<b>14 Wind instruments .....</b>	<b>383</b>
Changing the size of woodwind diagrams .....	383
Fingering symbols for wind instruments .....	383
Flute slap notation .....	384
Graphical and text woodwind diagrams .....	384
Recorder fingering chart .....	385
Woodwind diagrams key lists .....	386
Woodwind diagrams listing .....	387
<b>15 Chord notation .....</b>	<b>389</b>
Adding a figured bass above or below the notes .....	389
Adding bar lines to ChordNames context .....	389
Adjusting figured bass alteration glyphs .....	390
Changing a single note's size in a chord .....	390
Changing chord separator .....	390
Changing the positions of figured bass alterations .....	391
Chord name exceptions .....	391
Chord name major7 .....	392
Chord names alternative .....	392
Chords with stretched fingering for FretBoards and TabVoice .....	402
Clusters .....	403
Controlling the placement of chord fingerings .....	403
Cross-staff chords – beaming problems workaround .....	404
Customizing the chord grid style .....	404
Customizing the no-chord symbol .....	405
Display non-English chord names .....	406
Displaying complex chords .....	407
Manually break figured bass extenders for only some numbers .....	407
Print chord names with same root and different bass as slash and bass note .....	408
Showing chords at changes .....	410
Simple lead sheet .....	411
Single-staff template with notes, lyrics, and chords .....	411
Single-staff template with notes, lyrics, chords, and frets .....	412
Single-staff template with notes and chords .....	413
Vertically centering paired figured bass extenders .....	413
Volta below chords .....	414
<b>16 Contemporary music .....</b>	<b>415</b>
Beam nibs .....	415
Broken crescendo hairpin .....	416
Changing time signatures inside a polymetric section using \scaleDurations .....	416
Clusters .....	418
Contemporary glissando .....	418
Flat ties .....	419
Flute slap notation .....	421

Heavily customized polymetric time signatures .....	422
Marking notes of spoken parts with a cross on the stem (Sprechstimme) .....	423
Non-traditional key signatures .....	423
Printing music with different time signatures .....	424
Screech and Boink .....	426
Stemlets .....	428
<b>17 Ancient notation .....</b>	<b>429</b>
Adding a figured bass above or below the notes .....	429
Ancient fonts .....	429
Ancient notation template – modern transcription of Gregorian music .....	432
Ancient time signatures .....	433
Chant or psalm notation .....	433
Custodes .....	434
Incipit .....	435
Mensurstriche layout (bar lines between the staves) .....	439
Rest styles .....	440
Using tags to produce mensural and modern music from the same source .....	441
Vertical line as a baroque articulation mark .....	443
<b>18 World music .....</b>	<b>444</b>
Arabic improvisation .....	444
Makam example .....	444
Printing text from right to left .....	444
Turkish Makam example .....	445
<b>Other collections</b>	
<b>19 Automatic notation .....</b>	<b>449</b>
Automatic beam subdivisions .....	449
Forcing rehearsal marks to start from a given letter or number .....	449
Generating whole scores (also book parts) in Scheme without using the parser .....	450
Preventing extra naturals from being automatically added .....	452
Preventing natural signs from being printed when the key signature changes .....	452
Vocal ensemble template with automatic piano reduction .....	453
<b>20 Breaks .....</b>	<b>456</b>
Adding an extra staff at a line break .....	456
Positioning segno and coda (with line break) .....	457
Removing the first empty line .....	457
<b>21 Connecting notes .....</b>	<b>459</b>
Adding beams, slurs, ties, etc., when using tuplet and non-tuplet rhythms .....	459
Automatic beam subdivisions .....	459
Changing the appearance of a slur from solid to dotted or dashed .....	460
Controlling tuplet bracket visibility .....	460
Creating slurs across voices .....	461
Laissez vibrer ties .....	462
Manually controlling beam positions .....	462

<b>22</b>	<b>Contexts and engravers</b>	<b>463</b>
	Adding ambitus per voice	463
	Adding an extra staff	463
	Adding an extra staff at a line break	464
	Adding bar lines to ChordNames context	465
	Ambitus after key signature	465
	Analysis brackets with labels	465
	Automatically changing the stem direction of the middle note based on the melody	466
	Changing MIDI output to one channel per voice	467
	Changing time signatures inside a polymetric section using <code>\scaleDurations</code>	468
	Creating arpeggios across notes in different voices	469
	Creating blank staves	469
	Creating cross-staff arpeggios in other contexts	471
	Creating custom key signatures	472
	Cross-staff stems	473
	Defining an engraver in Scheme: ambitus engraver	473
	Displaying a whole GrandStaff system if only one of its staves is alive	480
	Engravers one by one	482
	Grid lines: changing their appearance	485
	Grid lines: emphasizing rhythms and notes synchronization	486
	Measure counters	487
	Measure spanner	488
	Mensurstriche layout (bar lines between the staves)	489
	Nesting staves	490
	Permitting line breaks within beamed tuplets	491
	Print chord names with same root and different bass as slash and bass note	492
	Printing marks on every staff	494
	Printing music with different time signatures	495
	Removing bar numbers from a score	497
	Use square bracket at the start of a staff group	498
	Using mark lines in a Frenched score	498
	Using tags to produce mensural and modern music from the same source	501
	Vocal ensemble template with verse and refrain	503
	Volta below chords	505
	Volta brackets in multiple staves	506
<b>23</b>	<b>Education</b>	<b>507</b>
	Grid lines: emphasizing rhythms and notes synchronization	507
	Making some staff lines thicker than the others	508
<b>24</b>	<b>Headword</b>	<b>509</b>
	Ancient headword	509
	Chords headword	511
	Editorial headword	513
	Expressive headword	515
	Figured bass headword	516
	Fretted headword	518
	Keyboard headword	521
	Pitches headword	525
	Repeats headword	526

Rhythms headword .....	528
Simultaneous headword .....	530
Staff headword .....	532
Text headword .....	534
Unfretted headword .....	536
Vocal headword .....	539
Wind headword .....	541
<b>25 MIDI .....</b>	<b>543</b>
Changing MIDI output to one channel per voice .....	543
Changing the tempo without a metronome mark .....	544
Creating custom dynamics in MIDI output .....	544
Demo of MIDI instruments .....	545
Replacing default MIDI instrument equalization .....	548
<b>26 Non-music .....</b>	<b>550</b>
Aligning and centering instrument names .....	550
Demonstrating all \header fields .....	551
Woodwind diagrams key lists .....	552
<b>27 Paper and layout .....</b>	<b>554</b>
Aligning and centering instrument names .....	554
Arranging separate lyrics on a single line .....	555
Book parts .....	556
Changing the staff size .....	560
Clip systems .....	561
Creating blank staves .....	562
Demonstrating all \header fields .....	564
Displaying a whole GrandStaff system if only one of its staves is alive .....	566
Setting system separators .....	567
Table of contents .....	568
Vertically aligned StaffGroups without connecting SystemStartBar .....	570
<b>28 Preparing parts .....</b>	<b>577</b>
Forcing rehearsal marks to start from a given letter or number .....	577
Numbering single measure rests .....	577
String quartet template with separate parts .....	577
<b>29 Real music .....</b>	<b>581</b>
Changing MIDI output to one channel per voice .....	581
Creating a sequence of notes on various pitches .....	582
Creating slurs across voices .....	582
Cross-staff tremolos .....	583
Demo of MIDI instruments .....	583
Dotted harmonics .....	587
Heavily customized polymetric time signatures .....	587
Indicating cross-staff chords with arpeggio bracket .....	588
Inserting score fragments above a staff, as markups .....	589
Percussion example .....	590
Printing music with different time signatures .....	591

<b>30 Really cool</b>	<b>594</b>
Adding the current date to a score	594
Blanking staff lines using the \whiteout command	594
Center text below hairpin dynamics	595
Changing properties for individual grobs	596
Clusters	597
Coloring notes depending on their pitch	597
Creating a sequence of notes on various pitches	598
Generating random notes	598
Generating whole scores (also book parts) in Scheme without using the parser	599
Making some staff lines thicker than the others	601
Non-traditional key signatures	601
Printing music with different time signatures	602
<b>31 Really simple</b>	<b>605</b>
Adding an extra staff	605
Adding drum parts	605
Adding fingerings to a score	606
Aligning text marks to notes	606
Analysis brackets above the staff	607
Changing a single note's size in a chord	607
Changing stanza fonts	608
Changing the appearance of a slur from solid to dotted or dashed	608
Combining dynamics with markup texts	609
Combining dynamics with markup texts (2)	609
Display non-English chord names	609
Forcing rehearsal marks to start from a given letter or number	611
Lyrics alignment	611
Merging multi-measure rests in a polyphonic part	612
Modifying tuplet bracket length	612
Outputting the version number	613
Piano template (simple)	613
Piano template with centered lyrics	614
Piano template with melody and lyrics	614
Single-staff template with notes, lyrics, and chords	615
Single-staff template with notes and chords	616
Single-staff template with notes and lyrics	617
Single-staff template with only notes	617
Skips in lyric mode	618
Skips in lyric mode (2)	618
String quartet template (simple)	618
Using the \tweak command to tweak individual grobs	620
Vocal ensemble template	620
Volta brackets in multiple staves	622
<b>32 Scheme</b>	<b>624</b>
Adding extra fingering with Scheme	624
Adding indicators to staves which get split after a break	624
Adding links to objects	628
Adding orchestral cues to a vocal score	630

Adding the current date to a score .....	632
Adjusting slur positions vertically .....	632
Center text below hairpin dynamics .....	633
Changing properties for individual grobs .....	635
Chord name exceptions .....	635
Coloring notes depending on their pitch .....	636
Creating “real” parenthesized dynamics .....	637
Creating a sequence of notes on various pitches .....	638
Creating custom dynamics in MIDI output .....	638
Customizing the position and number of dots in repeat sign bar lines .....	639
Defining an engraver in Scheme: ambitus engraver .....	640
Different font size settings for instrumentName and shortInstrumentName .....	647
Displaying grob ancestry .....	648
Drawing circles around note heads .....	650
Drawing circles around various objects .....	650
Extending glissandi across repeats .....	651
Flat ties .....	652
Flute slap notation .....	655
Fretboards alternate tables .....	655
Generate special note head shapes .....	657
Generating custom flags .....	658
Generating random notes .....	659
Generating whole scores (also book parts) in Scheme without using the parser .....	659
Isolated percent repeats .....	662
Numbers as easy note heads .....	662
Overriding articulations by type .....	663
Positioning grace notes with floating space .....	665
Print chord names with same root and different bass as slash and bass note .....	666
Replacing default MIDI instrument equalization .....	668
Separating key cancellations from key signature changes .....	669
String number extender lines .....	670
Three-sided box .....	671
Transposing pitches with minimum accidentals (“smart” transpose) .....	672
Two \partCombine pairs on one staff .....	673
User-defined time signatures .....	675
Using ly:grob-object to access grobs with \tweak .....	675
Vertical line as a baroque articulation mark .....	676
<b>33 Spacing .....</b>	<b>678</b>
Adjusting vertical spacing of lyrics .....	678
Allowing fingerings to be printed inside the staff .....	678
Breaking vertical alignment of dynamics and textscripts .....	679
Page label .....	679
Proportional strict notespacing .....	680
Reconciliating bar lines thickness when staves are of different sizes (TODO duplicates snippet in Documentation/en/notation/spacing.itely) .....	681
Vertically aligned dynamics and textscripts .....	683
Vertically aligning ossias and lyrics .....	684



<b>34</b>	<b>Specific notation</b>	<b>686</b>
	Accordion register symbols	686
	Adding bar lines to ChordNames context	686
	Adding drum parts	687
	Adding fingerings to tablatures	688
	Aiken head thin variant noteheads	688
	Allowing fingerings to be printed inside the staff	689
	Changing the number of lines in a staff	689
	Chant or psalm notation	690
	Chord name exceptions	690
	Chord name major7	691
	Chords with stretched fingering for FretBoards and TabVoice	691
	Clusters	692
	Contemporary glissando	692
	Controlling the placement of chord fingerings	693
	Cow and ride bell example	693
	Creating blank staves	694
	Custodes	696
	Demo of MIDI instruments	697
	Direction of merged ‘fa’ shape note heads	701
	Embedding native PostScript in a \markup block	701
	Engravers one by one	702
	Flamenco notation	704
	High and low woodblock example	708
	How to change fret diagram position	709
	How to put ties between syllables in lyrics	710
	Laissez vibrer ties	710
	Percussion example	710
	Tam-tam example	711
	Tambourine example	712
	Time signature in brackets	712
	Time signature in parentheses	712
	Using an extra voice for breaks	713
	Woodwind diagrams listing	714
<b>35</b>	<b>Symbols and glyphs</b>	<b>716</b>
	Accordion register symbols	716
	Adding indicators to staves which get split after a break	717
	Ancient fonts	721
	Breathing signs	724
	Broken crescendo hairpin	725
	Caesura (“railtracks”) with fermata	726
	Custodes	726
	Customizing the position and number of dots in repeat sign bar lines	727
	Fingering symbols for wind instruments	728
	How to put ties between syllables in lyrics	729
	Positioning segno and coda (with line break)	729
	Rest styles	730
	Volta text markup using repeatCommands	731

<b>36</b>	<b>Templates</b>	<b>732</b>
	Ancient notation template – modern transcription of Gregorian music	732
	Anglican psalm template	732
	Hymn template	735
	Jazz combo template	737
	Orchestra, choir and piano template	743
	Piano template (simple)	748
	Piano template with centered lyrics	748
	Piano template with melody and lyrics	749
	SATB choir template – four staves	750
	Single-staff template with notes, lyrics, and chords	752
	Single-staff template with notes, lyrics, chords, and frets	753
	Single-staff template with notes and chords	754
	Single-staff template with notes and lyrics	754
	Single-staff template with only notes	755
	String quartet template (simple)	755
	String quartet template with separate parts	757
	Vocal ensemble template	759
	Vocal ensemble template with automatic piano reduction	761
	Vocal ensemble template with lyrics aligned below and above the staves	763
	Vocal ensemble template with verse and refrain	765
<b>37</b>	<b>Titles</b>	<b>768</b>
	Adding the current date to a score	768
	Aligning and centering instrument names	768
	Demonstrating all \header fields	770
	Outputting the version number	771
<b>38</b>	<b>Tweaks and overrides</b>	<b>772</b>
	Adding an ottava marking to a single voice	772
	Adding links to objects	772
	Adding markups in a tablature	774
	Adding timing marks to long glissandi	775
	Adjusting grace note spacing	776
	Adjusting slur positions vertically	776
	Adjusting vertical spacing of lyrics	777
	Aligning text marks to notes	778
	Altering the length of beamed stems	778
	Alternative bar numbering	779
	Analysis brackets above the staff	780
	Analysis brackets with labels	780
	Asymmetric slurs	781
	Breaking vertical alignment of dynamics and textscripts	782
	Caesura (“railtracks”) with fermata	782
	Changing a single note’s size in a chord	783
	Changing beam thickness and spacing	783
	Changing form of multi-measure rests	783
	Changing properties for individual grobs	784
	Changing text and spanner styles for text dynamics	784
	Changing the default text font family	785

Changing the staff size .....	786
Changing the tempo without a metronome mark .....	786
Changing the text for sustain markings .....	787
Controlling spanner visibility after a line break .....	787
Controlling the appearance of tremolo slashes .....	788
Controlling the vertical ordering of scripts .....	789
Controlling tuplet bracket visibility .....	790
Creating a delayed turn .....	791
Creating custom key signatures .....	791
Creating text spanners .....	792
Cross-staff chords – beaming problems workaround .....	793
Cross-staff stems .....	793
Custodes .....	794
Customizing fretboard fret diagrams .....	795
Customizing markup fret diagrams .....	797
Display bracket with only one staff in a system .....	798
Displaying grob ancestry .....	799
Dotted harmonics .....	801
Drawing boxes around grobs .....	801
Drawing circles around various objects .....	801
Dynamics spanner with custom text .....	802
Extending a trill spanner .....	803
Extending glissandi across repeats .....	803
Fine-tuning pedal brackets .....	804
Flat ties .....	805
Force a cancellation natural before accidentals .....	807
Forcing horizontal shift of notes .....	808
Fret diagrams explained and developed .....	808
Generate special note head shapes .....	815
Generating custom flags .....	816
Glissandi can skip grobs .....	817
Hairpins with different line styles .....	817
Horizontally aligning custom dynamics like “più f” .....	818
How to change fret diagram position .....	821
Inserting a caesura .....	822
Keep change clefs full-sized .....	822
Line arrows .....	823
Making an object invisible using the ‘transparent’ property .....	823
Making glissandi breakable .....	824
Manually controlling beam positions .....	825
Measure-centered bar numbers .....	826
Mensurstriche layout (bar lines between the staves) .....	826
Modifying the ottava spanner slope .....	827
Moving dotted notes in polyphony .....	828
Moving slur positions vertically .....	828
Nesting staves .....	829
Overriding articulations by type .....	830
Percent repeat count visibility .....	831
Positioning arpeggios .....	832
Positioning fingering indications precisely .....	832
Positioning multi-measure rests .....	833
Positioning text markups inside slurs .....	834

Printing bar numbers inside boxes or circles .....	834
Printing metronome and rehearsal marks below the staff .....	835
Printing note names with and without an octave marker .....	835
Printing tuplet brackets on the note head side .....	836
Proportional strict notespacing .....	836
Removing brace on first line of piano score .....	837
Removing connecting bar lines on StaffGroup, PianoStaff, or GrandStaff .....	838
Removing the first empty line .....	838
Rest styles .....	840
Rhythmic slashes .....	841
Separating key cancellations from key signature changes .....	841
Setting hairpin behavior at bar lines .....	843
Setting system separators .....	843
Shape individual ties in chords .....	844
Showing the same articulation above and below a note or chord .....	847
String number extender lines .....	847
Suppressing warnings for clashing note columns .....	848
Time signature in brackets .....	848
Time signature in parentheses .....	849
Time signature printing only the numerator as a number (instead of the fraction) .....	849
Tuplet bracket and change staff .....	849
Tweaking clef properties .....	850
Tweaking grace layout within music .....	852
Using alternative flag styles .....	852
Using ly:grob-object to access grobs with \tweak .....	854
Using the \tweak command to tweak individual grobs .....	855
Vertically aligned dynamics and textscripts .....	855
Vertically aligning ossias and lyrics .....	856
Vertically aligning stanza numbers of different staves .....	857
Vertically centering paired figured bass extenders .....	858
 <b>39 Workaround .....</b>	 <b>860</b>
Adding an extra staff at a line break .....	860
Appoggiatura or grace note before a bar line .....	861
Breaking vertical alignment of dynamics and textscripts .....	861
Changing time signatures inside a polymetric section using \scaleDurations .....	862
Creating “real” parenthesized dynamics .....	863
Cross-staff chords – beaming problems workaround .....	864
Displaying complex chords .....	864
Extending glissandi across repeats .....	865
Forcing measure width to adapt to a metronome mark’s width .....	866
Making some staff lines thicker than the others .....	867
Marking notes of spoken parts with a cross on the stem (Sprechstimme) .....	867
Positioning grace notes with floating space .....	868
Positioning segno and coda (with line break) .....	869
Preventing final mark from removing final tuplet .....	870
Printing text from right to left .....	870
Transposing pitches with minimum accidentals (“smart” transpose) .....	871
Unfolding tremolo repeats .....	872
Using an extra voice for breaks .....	873
Vertically aligned dynamics and textscripts .....	874

## Preface

This document shows a selected set of LilyPond snippets from the LilyPond Wiki (<https://wiki.lilypond.community>), the successor of the LilyPond Snippet Repository (LSR).

We would like to address many thanks to Sebastiano Vigna for maintaining the LSR and Jean Abou Samra for maintaining the LilyPond Wiki!

Please note that this document is not an exact subset of the LilyPond Wiki: some snippets come from the Documentation/snippets/new/ LilyPond sources directory, and snippets from the Wiki are converted by `convert-ly`, as the LilyPond Wiki is (mainly) based on a stable LilyPond version, and this document is for version 2.25.32.

Snippets are grouped by LilyPond Wiki categories (<https://wiki.lilypond.community/wiki/Special:Categories>), using the same chapter order as the Notation Reference (if possible). Snippets may be tagged with several categories, which means that they may appear multiple times in this document. Not all Wiki categories may be present here, though.

For all snippets, the indentation of staves is set to zero.

In the HTML version of this document, you can click on the file name or figure for each example to see the corresponding input file.



## Musical notation





# 1 Pitches

See also Section “Pitches” in *Notation Reference*.

## Adding ambitus per voice

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

```
\new Staff <<
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c'' {
    \override Ambitus.X-offset = 2.0
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c' {
    \voiceTwo
    es4 f g as
    b1
  }
}>>
```



## Adding an ottava marking to a single voice

If you have more than one voice on the staff, setting octavation in one voice transposes the position of notes in all voices for the duration of the ottava bracket. If the octavation is only intended to apply to one voice, the *Ottava\_spanner\_engraver* should be moved to Voice context.

```
\layout {
  \context {
    \Staff
    \remove Ottava_spanner_engraver
  }
  \context {
    \Voice
    \consists Ottava_spanner_engraver
  }
}

{
  \clef bass
  << { <g d'>1~ q2 <c' e'> }
  \\\
  {
```

```

      r2.
      \ottava -1
      <b,,, b,,>4 ~ |
      q2
      \ottava 0
      <c e>2
    }
  >>
}

```



## Aiken head thin variant noteheads

Aiken head white notes get harder to read at smaller staff sizes, especially with ledger lines. Losing interior white space makes them appear as quarter notes.

```
\score {
{
    \aikenHeads
    c''2 a' c' a

    % Switch to thin-variant noteheads
    \set shapeNoteStyles = ##(doThin reThin miThin
                                faThin sol laThin tiThin)
    c'' a' c' a
}
}
```

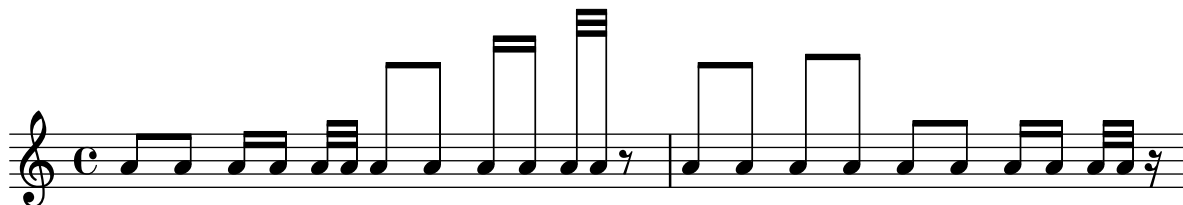


## Altering the length of beamed stems

Stem lengths on beamed notes can be varied by overriding the beamed-lengths property of the details of the Stem. If a single value is used as an argument, the length applies to all stems. When multiple arguments are used, the first applies to eighth notes, the second to sixteenth notes and so on. The final argument also applies to all notes shorter than the note length of the final argument. Non-integer arguments may also be used.

```
\relative c' {
  \override Stem.details.beamed-lengths = #'(2)
  a8[ a] a16[ a] a32[ a]
  \override Stem.details.beamed-lengths = #'(8 10 12)
  a8[ a] a16[ a] a32[ a] r8
  \override Stem.details.beamed-lengths = #'(8)
  a8[ a]
  \override Stem.details.beamed-lengths = #'(8.5)
  a8[ a]
```

```
\revert Stem.details.beamed-lengths
a8[ a] a16[ a] a32[ a] r16
}
```



## Ambitus

Ambitus indicate pitch ranges for voices.

Accidentals only show up if they are not part of the key signature. AmbitusNoteHead grobs also have ledger lines.

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

```
<<
\new Staff {
  \relative c' {
    \time 2/4
    c4 f'
  }
}
\new Staff {
  \relative c' {
    \time 2/4
    \key d \major
    cis4 as'
  }
}
>>
```



## Ambitus after key signature

By default, ambitus are positioned at the left of the clef. The `\ambitusAfter` function allows for changing this placement. Syntax is `\ambitusAfter grob-interface`; see Graphical Object Interfaces (<http://lilypond.org/doc/v2.24/Documentation/internals/graphical-object-interfaces>) for a list of possible values for `grob-interface`.

A common use case is printing the ambitus between key signature and time signature.

```
\new Staff \with {
  \consists Ambitus_engraver
} \relative {
  \ambitusAfter key-signature
  \key d \major
  es'8 g bes cis d2
}
```



## 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
  }
>>
```



## Applying note head styles depending on the step of the scale

The `shapeNoteStyles` property can be used to define various note head styles for each step of the scale (as set by the key signature or the tonic property).

This property requires a set of symbols, which can be purely arbitrary (geometrical expressions such as `triangle`, `cross`, and `xcircle` are allowed) or based on old American engraving tradition (some latin note names are also allowed).

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

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

```
fragment = {
  \key c \major
```

```

c2 d
e2 f
g2 a
b2 c
}

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

    \fragment
  }

  \break

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

    \fragment
  }
}

```



## 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.

The context property `suspendMelodyDecisions` may be used to turn off this behavior locally.

```

\relative c' {
  \time 3/4
  a8 b g f b g |
  \set suspendMelodyDecisions = ##t
  a b g f b g |
  \unset suspendMelodyDecisions
  c b d c b c |
}

\layout {
  \context {
    \Voice
    \consists "Melody_engraver"
    \autoBeamOff
  }
}

```

}



## Changing 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`.

Short text is especially useful when a brief ottava is used.

```
{
  c'2
  \ottava 1
  \set Staff.ottavation = "8"
  c''2
  \ottava 0
  c'1
  \ottava 1
  \set Staff.ottavation = "Text"
  c''1
}
```



## 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''
}

```



## Changing the interval of lines on the staff

`staffLineLayoutFunction` is used to change the position of notes. This snippet shows setting its value to `ly:pitch-semitones` in order to produce a chromatic scale with the distance between each space and line of the staff equal to one semitone.

```

scale = \relative c' {
  a4 ais b c
  cis4 d dis e
  f4 fis g gis
  a1
}

\new Staff \with {
  \remove "Accidental_engraver"
  staffLineLayoutFunction = #ly:pitch-semitones
}
{
  <<
    \scale
    \context NoteNames {
      \set printOctaveNames = ##f
      \scale
    }
  >>
}

```



## Clefs can be transposed by arbitrary amounts

Clefs can be transposed by arbitrary amounts, not just by octaves.

```
\relative c' {
  \clef treble
  c4 c c c
  \clef "treble_8"
  c4 c c c
  \clef "treble_5"
  c4 c c c
  \clef "treble^3"
  c4 c c c
}
```



## Coloring notes depending on their pitch

It is possible to color note heads depending on their pitch and/or their names: the function used in this example even makes it possible to distinguish enharmonics.

*% Association list of pitches to colors.*

```
#(define color-mapping
  (list
    (cons (ly:make-pitch 0 0 NATURAL) (x11-color 'red))
    (cons (ly:make-pitch 0 0 SHARP) (x11-color 'green))
    (cons (ly:make-pitch 0 1 FLAT) (x11-color 'green))
    (cons (ly:make-pitch 0 2 NATURAL) (x11-color 'red))
    (cons (ly:make-pitch 0 2 SHARP) (x11-color 'green))
    (cons (ly:make-pitch 0 3 FLAT) (x11-color 'red))
    (cons (ly:make-pitch 0 3 NATURAL) (x11-color 'green))
    (cons (ly:make-pitch 0 4 SHARP) (x11-color 'red))
    (cons (ly:make-pitch 0 5 NATURAL) (x11-color 'green))
    (cons (ly:make-pitch 0 5 FLAT) (x11-color 'red))
    (cons (ly:make-pitch 0 6 SHARP) (x11-color 'red))
    (cons (ly:make-pitch 0 1 NATURAL) (x11-color 'blue))
    (cons (ly:make-pitch 0 3 SHARP) (x11-color 'blue))
    (cons (ly:make-pitch 0 4 FLAT) (x11-color 'blue))
    (cons (ly:make-pitch 0 5 SHARP) (x11-color 'blue))
    (cons (ly:make-pitch 0 6 FLAT) (x11-color 'blue))))
```

*% Compare pitch and alteration (not octave).*

```
#(define (pitch-equals? p1 p2)
  (and
    (= (ly:pitch-alteration p1) (ly:pitch-alteration p2))
    (= (ly:pitch-notename p1) (ly:pitch-notename p2))))
```



```

#(define (pitch-to-color pitch)
  (let ((color (assoc pitch color-mapping pitch-equals?)))
    (if color
        (cdr color))))

#(define (color-notehead grob)
  (pitch-to-color
   (ly:event-property (event-cause grob) 'pitch)))

\score {
  \new Staff \relative c' {
    \override NoteHead.color = #color-notehead
    c8 b d dis ees f g aes
  }
}

```



## Creating a sequence of notes on various pitches

In music that contains many occurrences of the same sequence of notes at different pitches, the following music function may prove useful. It takes a note, of which only the pitch is used.

This example creates the rhythm used throughout *Mars*, from Gustav Holst's *The Planets*.

```

rhythm =
#(define-music-function (p) (ly:pitch?)
  "Make the rhythm in Mars (the Planets) at the given pitch"
  #{ \tuplet 3/2 { $p 8 8 8 } 4 4 8 8 4 #})

\new Staff {
  \time 5/4
  \rhythm c'
  \rhythm c''
  \rhythm g
}

```



## Creating custom key signatures

LilyPond supports custom key signatures. In this example, print for D minor and D major with an extended range of shown flats.

```

\new Staff \with {
  \override StaffSymbol.line-count = #8
  \override KeySignature.flat-positions = #'((-7 . 6))
  \override KeyCancellation.flat-positions = #'((-7 . 6))
  \override KeySignature.sharp-positions = #'((-6 . 7))
  \override KeyCancellation.sharp-positions = #'((-6 . 7))
}

```

```

\override Clef.stencil =
  #(\lambda (grob)
    (grob-interpret-markup grob
      #{ \markup\combine
        \musicglyph "clefs.C"
        \translate #'(-3 . -2)
        \musicglyph "clefs.F"
      })
    clefPosition = #3
    middleCPosition = #3
    middleCClefPosition = #3
  )
}

{
  \key d\minor f bes, f bes, |
  \key d\major fis b, fis b, |
}

```



## Direction of merged ‘fa’ shape note heads

Using property `NoteCollision.fa-merge-direction`, the direction of “fa” shape note heads (“fa”, “faThin”, etc.) can be controlled independently of the stem direction if two voices with the same pitch and different stem directions are merged. If this property is not set, the “down” glyph variant is used.

```

{
  \clef bass

  << { \aikenHeads
    f2
    \override Staff.NoteCollision.fa-merge-direction = #UP
    f2 }
  \\ { \aikenHeads
    f2
    f2 }
  >>
}

```



## Force a cancellation natural before accidentals

The following example shows how to force a natural sign before an accidental.

```

\relative c' {
  \key es \major
  bes c des
  \tweak Accidental.restore-first ##t
}

```

```
eis
}
```



## Forcing a clef symbol to be displayed

When a clef sign has already been displayed and it has not been changed to a different clef, then repeating the `\clef` command will be ignored by LilyPond, since it is not a change of clef. It is possible to force the clef to be redisplayed using the command `\set Staff.forceClef = ##t`.

```
\relative c' {
  \clef treble
  c1
  \clef treble
  c1
  \set Staff.forceClef = ##t
  c1
  \clef treble
  c1
}
```



## Generating random notes

This Scheme-based snippet generates random notes. Use as

```
\randomNotes n from to dur
```

to generate *n* random notes between pitches *from* and *to*, with duration *dur*.

```
randomNotes =
#(define-music-function (n from to dur)
  (integer? ly:pitch? ly:pitch? ly:duration?)
  (let ((from-step (ly:pitch-steps from))
        (to-step (ly:pitch-steps to)))
    (make-sequential-music
     (map (lambda (_)
            (let* ((step (+ from-step
                              (random (- to-step from-step))))
                  (pitch (ly:make-pitch 0 step 0)))
              #{ $pitch $dur #}))
          (iota n))))))
```

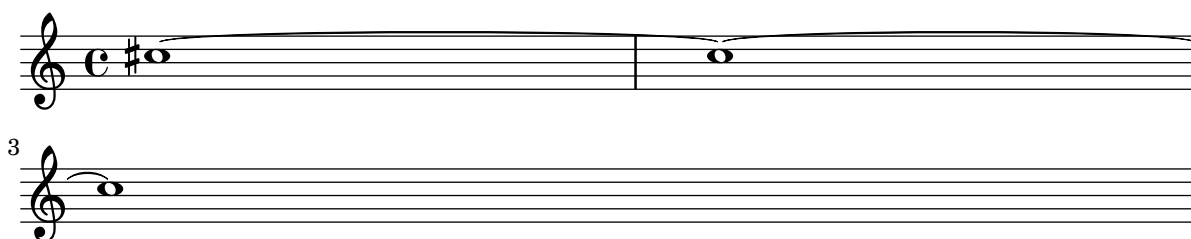
```
\randomNotes 24 c' g'' 8
```



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

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

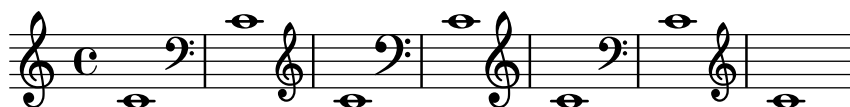
```
\relative c' {
  \override Accidental.hide-tied-accidental-after-break = ##t
  cis1~ cis~
  \break
  cis
}
```



## Keep change clefs full-sized

When a clef changes, the clef sign displayed is smaller than the initial clef. This can be overridden by setting the context property `full-size-change` to `##t`.

```
\relative c' {
  \clef "treble"
  c1
  \clef "bass"
  c1
  \clef "treble"
  c1
  \override Staff.Clef.full-size-change = ##t
  \clef "bass"
  c1
  \clef "treble"
  c1
  \revert Staff.Clef.full-size-change
  \clef "bass"
  c1
  \clef "treble"
  c1
}
```



## Makam example

Makam is a type of melody from Turkey using 1/9-tone microtonal alterations.

Consult the initialization file `ly/makam.ly` for details of pitch names and alterations.

```
\include "makam.ly"
```

```
\relative c' {
  \set Staff.keyAlterations = #`((6 . , (- KOMA)) (3 . ,BAKIYE))
```

```

c4 cc db fk
gbm4 gfc gfb efk
fk4 db cc c
}

```



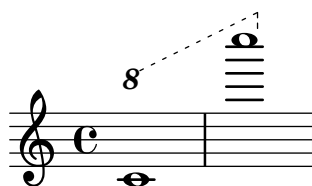
## Modifying the ottava spanner slope

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

```

\relative c'' {
  \override Staff.OttavaBracket.stencil = #ly:line-spanner::print
  \override Staff.OttavaBracket.bound-details =
    #`((left . ((Y . 0)
      (attach-dir . ,LEFT)
      (padding . 0)
      (stencil-align-dir-y . ,CENTER)))
      (right . ((Y . 5.0) ; Change the number here
        (padding . 0)
        (attach-dir . ,RIGHT)
        (text . ,(make-draw-dashed-line-markup
          (cons 0 -1.2))))))
  \override Staff.OttavaBracket.left-bound-info =
    #ly:horizontal-line-spanner::calc-left-bound-info-and-text
  \override Staff.OttavaBracket.right-bound-info =
    #ly:horizontal-line-spanner::calc-right-bound-info
  \ottava 1
  c1
  c''''1
}

```



## Non-traditional key signatures

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

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

```

\set Staff.keyAlterations =
  #`(((octave . step) . alter) ((octave . step) . alter) ...)

```

where, for each element in the list, *octave* specifies the octave (0 being the octave from middle C to the B above), *step* specifies the note within the octave (0 means C and 6 means B), and *alter* is one of SHARP, FLAT, DOUBLE-SHARP, etc., preceded by a comma.

Alternatively, you can use the more concise format *(step . alter)* for each item in the list if the same alterations are used 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.

```
\include "arabic.ly"
```

```
\relative do' {
  \set Staff.keyAlterations = #`((0 . ,SEMI-FLAT)
                                (1 . ,SEMI-FLAT)
                                (2 . ,FLAT)
                                (5 . ,FLAT)
                                (6 . ,SEMI-FLAT))

  % \set Staff.extraNatural = ##f
  re reb \down reb resd
  dod dob dosd \down dob |
  dobsb dodsdo do do |
}
```



## Numbers as easy note heads

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

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

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

#(set-global-staff-size 30)

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

\relative c' {
```

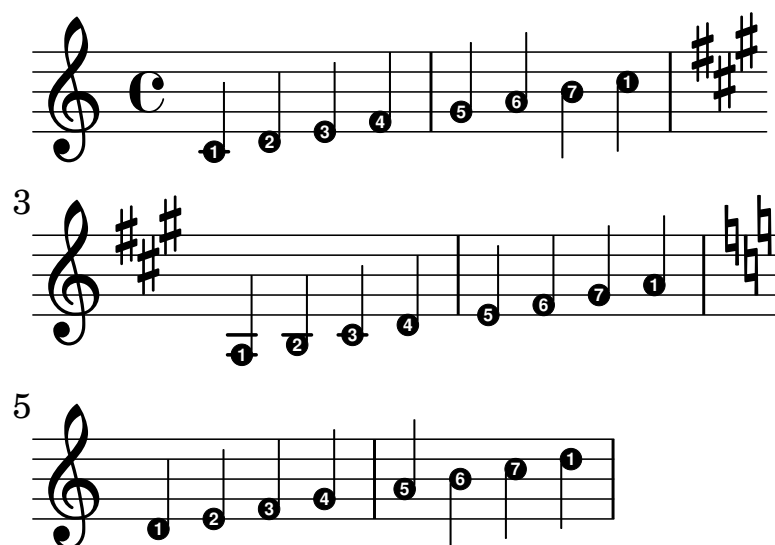
```

\easyHeadsOn
c4 d e f
g4 a b c \break

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

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

```



## Orchestra, choir and piano template

This template demonstrates the use of nested `StaffGroup` and `GrandStaff` contexts to subgroup instruments of the same type together, and a way to use `\transpose` so that variables hold music for transposing instruments at concert pitch.

```

#(set-global-staff-size 17)

```

```

\paper {
  indent = 3.0\cm % add space for instrumentName
  short-indent = 1.5\cm % add less space for shortInstrumentName
}

```

```

fluteMusic = \relative c' { \key g \major g'1 b }

```

```

% Pitches as written on a manuscript for Clarinet in A
% are transposed to concert pitch.

```

```

clarinetMusic = \transpose c' a
  \relative c'' { \key bes \major bes1 d }

```

```

trumpetMusic = \relative c { \key g \major g''1 b }

```

```

% Key signature is often omitted for horns

```

```

hornMusic = \transpose c' f

```

```

\relative c { d'1 fis }

percussionMusic = \relative c { \key g \major g1 b }

sopranoMusic = \relative c' { \key g \major g'1 b }
sopranoLyrics = \lyricmode { Lyr -- ics }

altoIMusic = \relative c' { \key g \major g'1 b }
altoILyrics = \sopranoLyrics
altoIIMusic = \relative c' { \key g \major g'1 b }
altoIILyrics = \lyricmode { Ah -- ah }

tenorMusic = \relative c' { \clef "treble_8" \key g \major g1 b }
tenorLyrics = \sopranoLyrics

pianoRHMus = \relative c { \key g \major g''1 b }
pianoLHMus = \relative c { \clef bass \key g \major g1 b }

violinIMusic = \relative c' { \key g \major g'1 b }
violinIIMusic = \relative c' { \key g \major g'1 b }

violaMusic = \relative c { \clef alto \key g \major g'1 b }

celloMusic = \relative c { \clef bass \key g \major g1 b }

bassMusic = \relative c { \clef "bass_8" \key g \major g,1 b }

\book {
  \score {
    <<
    \new StaffGroup = "StaffGroup_woodwinds" <<
      \new Staff = "Staff_flute" \with { instrumentName = "Flute" }
        \fluteMusic

      \new Staff = "Staff_clarinet" \with {
        instrumentName = \markup { \concat { "Clarinet in B" \flat } }
      }
      % Declare that written Middle C in the music
      % to follow sounds a concert B flat, for
      % output using sounded pitches such as MIDI.
      %\transposition bes

      % Print music for a B-flat clarinet
      \transpose bes c' \clarinetMusic
    >>

    \new StaffGroup = "StaffGroup_brass" <<
      \new Staff = "Staff_hornI" \with {
        instrumentName = "Horn in F"
      }
      % \transposition f
      \transpose f c' \hornMusic
  }
}

```



```

\new Staff = "Staff_trumpet" \with {
  instrumentName = "Trumpet in C"
}
\trumpetMusic
>>

\new RhythmicStaff = "RhythmicStaff_percussion" \with {
  instrumentName = "Percussion"
}
\percussionMusic

\new PianoStaff \with {
  instrumentName = "Piano"
} <<
  \new Staff { \pianoRHMusical }
  \new Staff { \pianoLHMusical }
>>

\new ChoirStaff = "ChoirStaff_choir" <<
  \new Staff = "Staff_soprano" \with {
    instrumentName = "Soprano"
  }
  \new Voice = "soprano" \sopranoMusical
  \new Lyrics \lyricsto "soprano" { \sopranoLyrics }

  \new GrandStaff = "GrandStaff_alto" \with {
    \accepts Lyrics
  } <<
    \new Staff = "Staff_altoI" \with {
      instrumentName = "Alto I"
    }
    \new Voice = "altoI"
    \altoIMusical
    \new Lyrics \lyricsto "altoI" { \altoILyrics }
    \new Staff = "Staff_altoII" \with {
      instrumentName = "Alto II"
    }
    \new Voice = "altoII"
    \altoIIMusical
    \new Lyrics \lyricsto "altoII" { \altoIILyrics }
  >>

  \new Staff = "Staff_tenor" \with {
    instrumentName = "Tenor"
  }
  \new Voice = "tenor" \tenorMusical
  \new Lyrics \lyricsto "tenor" { \tenorLyrics }
>>

\new StaffGroup = "StaffGroup_strings" <<
  \new GrandStaff = "GrandStaff_violins" <<

```

```
\new Staff = "Staff_violinI" \with {  
  instrumentName = "Violin I"  
}  
  \violinIMusic  
\new Staff = "Staff_violinII" \with {  
  instrumentName = "Violin II"  
}  
  \violinIIMusic  
>>  
  
\new Staff = "Staff_viola" \with {  
  instrumentName = "Viola"  
}  
  \violaMusic  
  
\new Staff = "Staff_cello" \with {  
  instrumentName = "Cello"  
}  
  \celloMusic  
  
\new Staff = "Staff_bass" \with {  
  instrumentName = "Double Bass"  
}  
  \bassMusic  
>>  
>>  
}  
}
```

Flute

Clarinet in B $\flat$

Horn in F

Trumpet in C

Percussion

Piano

Soprano

Alto I

Alto II

Tenor

Violin I

Violin II

Viola

Cello

Double Bass

Lyr - ics

Lyr - ics

Ah - ah

Lyr - ics

## Preventing extra naturals from being automatically added

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

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



## Preventing natural signs from being printed when the key signature changes

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

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



## Quoting another voice with transposition

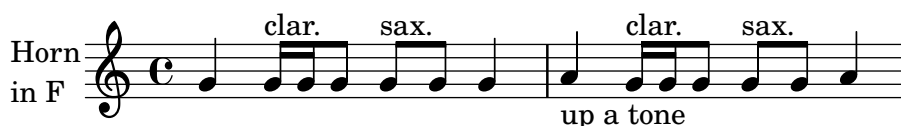
Quotations take into account the transposition of both source and target. In this example, all instruments play sounding middle c; the target is an instrument in f. The target part may be transposed using `\transpose`. In this case, all the pitches (including the quoted ones) are transposed.

```
\addQuote clarinet {
  \transpose bes
  \repeat unfold 8 { d'16 d' d'8 }
}

\addQuote sax {
  \transpose es'
  \repeat unfold 16 { a8 }
}
```

```
quoteTest = {
  % french horn
  \transposition f
  g'4
  << \quoteDuring "clarinet" { \skip 4 } s4^"clar." >>
  << \quoteDuring "sax" { \skip 4 } s4^"sax." >>
  g'4
}

{
  \new Staff \with {
    instrumentName = \markup { \column { Horn "in F" } }
  }
  \quoteTest
  \transpose c' d' << \quoteTest s4_"up a tone" >>
}
```



## Separating key cancellations from key signature changes

By default, the accidentals used for key cancellations are placed adjacent to those for key signature changes. This behavior can be changed by overriding the `break-align-orders` property of the `BreakAlignment` grob.

The value of `break-align-orders` is a vector of length 3, with quoted lists of breakable items as elements. Each list describes the default order of prefatory matter at the end, in the middle, and at the beginning of a line, respectively. We are only interested in changing the behaviour in the middle of a line.

If you look up the definition of `break-align-orders` in LilyPond's Internal Reference (see the `BreakAlignment` (<https://lilypond.org/doc/v2.24/Documentation/internals/breakalignment>) grob), you get the following order in the second element:

```
...
staff-bar
key-cancellation
key-signature
...
```

We want to change that, moving key-cancellation before staff-bar. To make this happen we use the `grob-transformer` function, which gives us access to the original vector as the second argument of the lambda function, here called *orig* (we don't need the first argument, *grob*). We return a new vector, with unchanged first and last elements. For the middle element, we first remove key-cancellation from the list, then adding it again before staff-bar.

```
#(define (insert-before where what lst)
  (cond
    ((null? lst) ; If the list is empty,
     (list what)) ; return a single-element list.
    ((eq? where (car lst)) ; If we find symbol `where`,
     (cons what lst)) ; insert `what` before curr. position.
    (else ; Otherwise keep building the list by
     (cons (car lst) ; adding the current element and
```

```

                                ; recursing with the next element.
                                (insert-before where what (cdr lst))))))

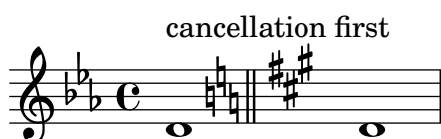
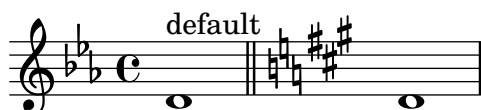
cancellationFirst =
\override Score.BreakAlignment.break-align-orders =
#(grob-transformer
  'break-align-orders
  (lambda (grob orig)
    (let* ((middle (vector-ref orig 1))
           (middle (delq 'key-cancellation middle))
           (middle (insert-before
                        'staff-bar 'key-cancellation middle)))
      (vector
        ;; end of line
        (vector-ref orig 0)
        ;; middle of line
        middle
        ;; beginning of line
        (vector-ref orig 2))))))

music = { \key es \major d'1 \bar "||"
          \key a \major d'1 }

{ <>^\markup "default"
  \music }

{ <>^\markup "cancellation first"
  \cancellationFirst
  \music }

```



## 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))
      ;; `ly:pitch-alteration` returns quarter tone steps.
      (a (* 4 (ly:pitch-alteration p)))
      (n (ly:pitch-notename p)))
  (cond
    ((and (> a 1)
          (or (eqv? n 6) (eqv? n 2))))
    (set! a (- a 2))
    (set! n (+ n 1)))
    ((and (< a -1)
          (or (eqv? n 0) (eqv? 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))))
  (when (< n 0)
    (set! o (- o 1))
    (set! n (+ n 7)))
  (when (> n 6)
    (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)))
    (when (pair? es)
      (ly:music-set-property! music 'elements
                              (map naturalize es)))
    (when (ly:music? e)
      (ly:music-set-property! music 'element
                              (naturalize e)))
    (when (ly:pitch? p)
      (set! p (naturalize-pitch p))
      (ly:music-set-property! music 'pitch p))
    music))

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

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

\new Staff {
  \transpose c ais { \music }
  \naturalizeMusic \transpose c ais { \music }

```

```
\transpose c deses { \music }
\naturalizeMusic \transpose c deses { \music }
}
```



## Turkish Makam example

This template uses the start of a well-known Turkish *Saz Semai* that is familiar in the repertoire in order to illustrate some of the elements of Turkish music notation.

```
##(set-default-paper-size "a6" 'landscape)
```

```
\include "turkish-makam.ly"
```

```
\header {
  title = "Hüseyin Saz Semaisi"
  composer = "Lavtacı Andon"
  tagline = ##f
}
```

```
\relative {
  \set Staff.extraNatural = ##f
  \set Staff.autoBeaming = ##f
```

```
\key a \huseyni
\time 10/8
```

```
a'4 g'16[ fb] e8.[ d16] d[ c d e] c[ d c8] bfc |
a16[ bfc a8] bfc c16[ d c8] d16[ e d8] e4 fb8 |
d4 a'8 a16[ g fb e] fb8[ g] a8.[ b16] a16[ g] |
g4 g16[ fb] fb8.[ e16] e[ g fb e] e4 r8 |
```

```
}
```

```
\layout {
  indent = 0
}
```

## Hüseyin Saz Semaisi

Lavtacı Andon





## Tweaking clef properties

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

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

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

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

  % The following clef changes do not preserve
  % the normal relationship between notes, key signatures
  % and clefs.
  \set Staff.clefGlyph = "clefs.F"
  \set Staff.clefPosition = 2
```

```

c'1
\set Staff.clefGlyph = "clefs.G"
c'1
\set Staff.clefGlyph = "clefs.C"
c'1
\set Staff.clefTransposition = 7
c'1
\set Staff.clefTransposition = 0
\set Staff.clefPosition = 0
c'1

% Return to the normal clef.
\set Staff.middleCPosition = 0
c'1
}

```



## Using \autoChange with more than one voice

Here is a demonstration of how to use \autoChange with more than one voice.

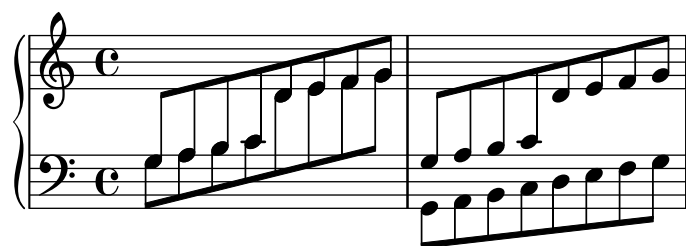
```

\score {
  \new PianoStaff
  <<
    \new Staff = "up" {
      <<
        \set Timing.beamExceptions = #'()
        \set Timing.beatStructure = #'(4)
        \new Voice {
          \voiceOne
          \autoChange
          \relative c' {
            g8 a b c d e f g
            g,8 a b c d e f g
          }
        }

        \new Voice {
          \voiceTwo
          \autoChange
          \relative c' {
            g8 a b c d e f g
            g,,8 a b c d e f g
          }
        }
      >>
    }
  >>
}

```

```
\new Staff = "down" {  
  \clef bass  
}  
>>  
}
```



## 2 Rhythms

See also Section “Rhythms” in *Notation Reference*.

### Adding beams, slurs, ties, etc., when using tuplet and non-tuplet rhythms

LilyPond primarily uses postfix syntax for inputting parentheses, brackets, etc., which might feel unintuitive for novices.

For example, when entering a manual beam, the left square bracket has to be placed *after* the starting note and its duration, not before. Similarly, the right square bracket should directly follow the note which is to be at the end of the requested beaming, even if this note happens to be inside a tuplet section.

This snippet demonstrates how to combine manual beaming, manual slurs, ties, and phrasing slurs with tuplet sections (enclosed within curly braces).

```
{
  r16[ g16 \tuplet 3/2 { r16 e'8] }
  g16( a \tuplet 3/2 { b d' e' } )
  g8[( a \tuplet 3/2 { b d' ) e']\ ( ~ }
  \time 2/4
  \tuplet 5/4 { e'32 a b d' e' } a'4.\)
}
```



### Adding drum parts

Using the powerful pre-configured tools such as the `\drummode` function and the `DrumStaff` context, inputting drum parts is quite easy: drums are placed at their own staff positions (with a special clef symbol) and have note heads according to the drum. Attaching an extra symbol to the drum or restricting the number of lines is possible.

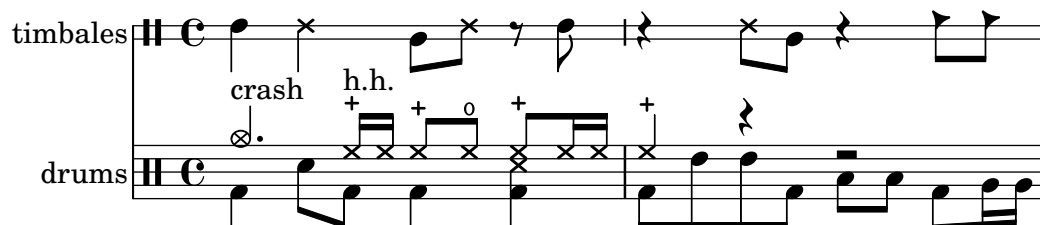
```
drh = \drummode {
  cymc4.^"crash" hhc16^"h.h." hh hhc8 hho hhc8 hh16 hh
  hhc4 r4 r2
}
drl = \drummode {
  bd4 sn8 bd bd4 << bd ss >>
  bd8 tommh tommh bd toml toml bd tomfh16 tomfh
}
timb = \drummode {
  timh4 ssh timl8 ssh r timh r4
  ssh8 timl r4 cb8 cb
}

\score {
  <<
  \new DrumStaff \with {
    instrumentName = "timbales"
    drumStyleTable = #timbales-style
```

```

\override StaffSymbol.line-count = #2
\override BarLine.bar-extent = #'(-1 . 1)
}
<<
\timb
>>
\new DrumStaff \with { instrumentName = "drums" }
<<
\new DrumVoice { \stemUp \drh }
\new DrumVoice { \stemDown \drl }
>>
>>
\layout { }
\midi { \tempo 4 = 120 }
}

```



## Adjusting grace note spacing

The space given to grace notes can be adjusted using the spacing-increment property of `Score.GraceSpacing`.

```

graceNotes = {
  \grace { c4 c8 c16 c32 }
  c8
}

\relative c' {
  c8
  \graceNotes
  \override Score.GraceSpacing.spacing-increment = #2.0
  \graceNotes
  \revert Score.GraceSpacing.spacing-increment
  \graceNotes
}

```



## Aligning bar numbers

The default alignment of bar numbers depends on its position: at the beginning of a staff, bar numbers are right-aligned; at all other positions, they are left-aligned. Using Scheme function `break-alignment-list`, this can be changed; the three arguments of this function are the alignment for end-of-line, middle-of-line, and start-of-line position (in this order).

```

\relative c' {

```

```

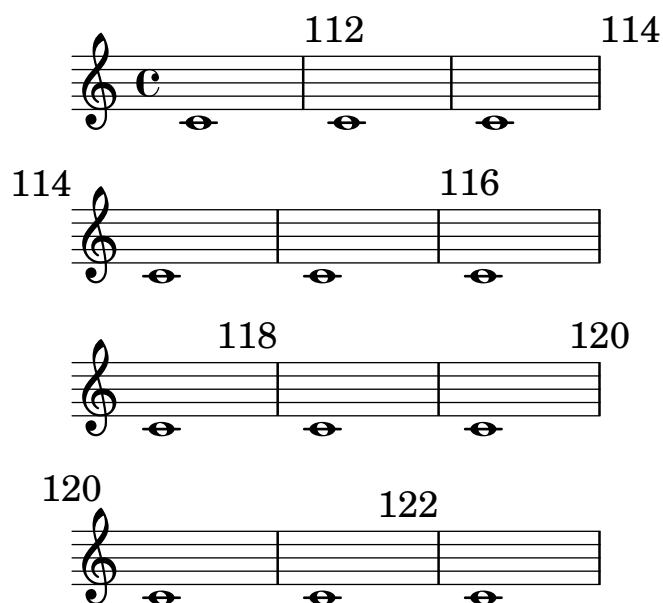
\set Score.currentBarNumber = 111
\override Score.BarNumber.break-visibility = #all-visible
% Increase the size of the bar number by 2
\override Score.BarNumber.font-size = 2
% Print a bar number every second measure
\set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)

c1 | c1 | c1 | \break
c1 | c1 | c1 | \break

\override Score.BarNumber.self-alignment-X =
  #(break-alignment-list CENTER RIGHT CENTER)
c1 | c1 | c1 | \break
c1 | c1 | c1 |
}

\paper {
  line-width = 70\mm
}

```



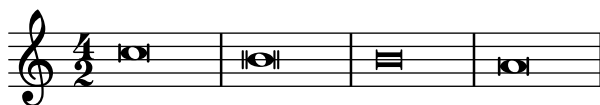
## Alternative breve notes

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

```

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

```



## Appoggiatura or grace note before a bar line

By default, appoggiaturas and grace notes that occur on the first beat of a measure are printed after the bar line. A possible solution for single staves to print it before the bar line is to add an invisible bar line and then the visible one.

In multi-staff systems, however, adding an invisible bar line distorts the positioning of full-bar rests in other staves; they are no longer centered but slightly shifted to the left. A better solution for such situations is to use the `\afterGrace` command with setting `afterGraceFraction` appropriately.

```
<<
{
  \appoggiatura d''8 c''4 r2. |
  \appoggiatura { \bar "" d''8 \bar "|" } |
  c''4 r2.
}
{ R1 | R1 }
>>
```

`afterGraceFraction = 15/16`

```
<<
{
  \appoggiatura d''8 c''4 \afterGrace r2. d''8( |
  c''4) r2.
}
{ R1 | R1 }
>>
```



## Automatic beam subdivisions

Beams can be subdivided automatically. By setting the property `subdivideBeams`, beams are subdivided whenever possible. The intervals and depth of subdivision can be limited with properties `beamMinimumSubdivision` and `beamMaximumSubdivision`, respectively.

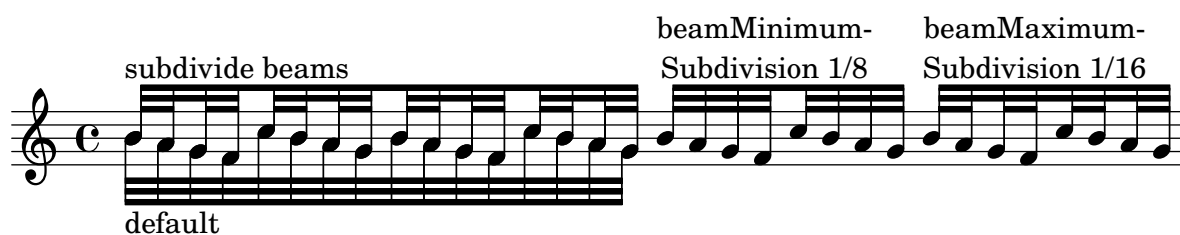
```
\new Staff {
```

```

\relative c' {
  <<
  {
    \voiceOne
    \set subdivideBeams = ##t
    b32["subdivide beams" a g f c' b a g
    b32 a g f c' b a g]
  }
  \new Voice {
    \voiceTwo
    b32_"default"[ a g f c' b a g
    b32 a g f c' b a g]
  }
  >>
  \oneVoice
  \once \set beamMinimumSubdivision = #1/8
  b32^\markup \center-column { "beamMinimum-"
    "Subdivision 1/8" } [ a g f c' b a g]

  \once \set beamMaximumSubdivision = #1/16
  b32^\markup \center-column { "beamMaximum-"
    "Subdivision 1/16" } [ a g f c' b a g]
}
}

```



## Automatically change durations

`shiftDurations` can be used to change the note lengths of a piece of music.

It takes two arguments – the scaling factor as a power of two, and the number of dots to be added as a positive integer.

```
music = \relative c' { a1 b2 c4 d8 r }
```

```

{
  \time 4/2
  \music
  \time 4/4
  \shiftDurations 1 0 \music
  \time 2/4
  \shiftDurations 2 0 \music
  \time 4/1
  \shiftDurations -1 0 \music
  \time 8/1
  \shiftDurations -2 0 \music
  \time 6/2
  \shiftDurations 0 1 \music
  \time 7/2
}

```



```
\shiftDurations 0 2 \music
}
```



## 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.beatBase = #1/8
  \set Score.beatStructure = 3,4,3
  <<
    \new Staff {
      c8 c c c c c c c c c
    }
    \new Staff {
      % Modify beaming for just this staff
      \set Staff.beatStructure = 6,4
      c8 c c c c c c c c c
    }
    \new Staff {
      % Inherit beaming from Score context
      <<
        {
          \voiceOne
          c8 c c c c c c c c c
        }
        % Modify beaming for this voice only
        \new Voice {
          \voiceTwo
          \set Voice.beatStructure = 6,4
          a8 a a a a a a a a
        }
      >>
    }
  >>
}
```



## Beam nibs

Beam nibs at the start and end of beams together with beams attached to solitary notes that look like flat flags are possible with a combination of `stemLeftBeamCount`, `stemRightBeamCount`, and paired `[]` beam indicators.

For imitating right-pointing flat flags on lone notes, use paired `[]` beam indicators and set `stemLeftBeamCount` to zero. For imitating left-pointing flat flags on lone notes, set `stemRightBeamCount` to zero instead (line one).

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

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

Note that `\set stemLeftBeamCount` is always equivalent to `\once \set`. In other words, the beam count settings are not “sticky”, so the pair of nibs attached to the lone 16th note in the last example has nothing to do with the `\set` command for the beam before.

```
\score {
  <<
    \new RhythmicStaff {
      \set stemLeftBeamCount = 0
      c16[] r8.
      r8.
      \set stemRightBeamCount = 0
      16[]
    }
    \new RhythmicStaff {
      16 16
      \set stemRightBeamCount = 2
      16 r r
      \set stemLeftBeamCount = 2
      16 16 16
    }
    \new RhythmicStaff {
      16 16
      \set stemRightBeamCount = 2
      16 r16
      16[] r16
      \set stemLeftBeamCount = 2
      16 16
    }
  }
}
```



## Beams across line breaks

Normally, LilyPond refuses to automatically break a line at places where a beam crosses a bar line. This behavior can be changed by setting the `Beam.breakable` property to `#t`.

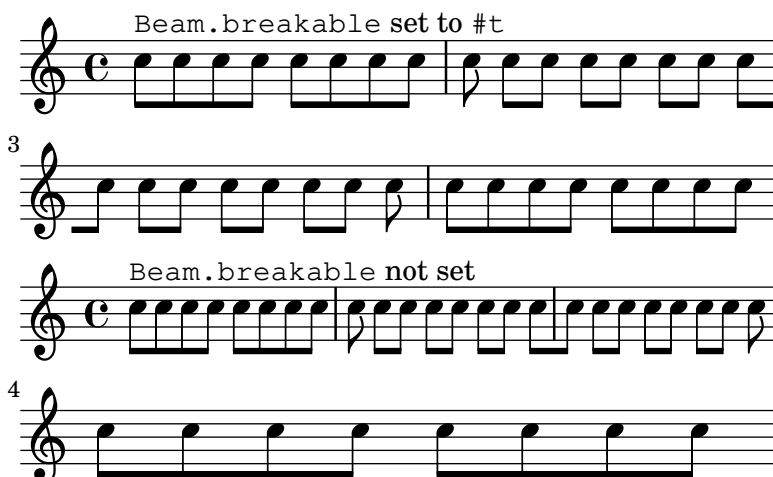
This property does not affect manual breaks inserted with commands like `\break`.

```
music = {
  \repeat unfold 8 c8
  c8 \repeat unfold 7 { c[ c] } c
  \repeat unfold 8 c8
}

\relative c'' {
  <>\markup { \typewriter Beam.breakable set to \typewriter "#t" }
  \override Beam.breakable = ##t
  \music
}

\relative c'' {
  <>\markup { \typewriter Beam.breakable not set }
  \music
}

\paper {
  line-width = 100\mm
}
```



## Changing beam knee gap

Kneaded 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 kneaded beam is drawn if the gap is larger than the value of `auto-knee-gap` plus the width of the beam object (which depends on the duration of the notes and the slope of the beam). By default, `auto-knee-gap` is set to 5.5 staff spaces.

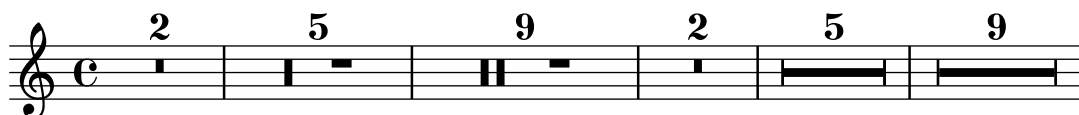
```
{
  f8 f''8 f8 f''8
  \override Beam.auto-knee-gap = #6
  f8 f''8 f8 f''8
}
```



## 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 long, thick horizontal line is shown. This default value of ten may be changed by overriding the `expand-limit` property.

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



## Changing the number of augmentation dots per note

The number of augmentation dots on a single note can be overridden by setting the `dot-count` property of the `Dots` grob.

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



## 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 { }
}
```



## Changing the tuplet number

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

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

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



## Changing time signatures inside a polymetric section using `\scaleDurations`

### *Flexible polymeter with unaligned measures*

To support explicit creation of independently measured contexts, remove the `Timing_translator` from `Score` context and define a `TimingStaffGroup` context that has `Timing_translator`. This makes `Timing` an alias for `TimingStaffGroup`, targeting `\time` commands to the enclosing `TimingStaffGroup`.

Unlike LilyPond's built-in `\enablePerStaffTiming` command, this approach requires the explicit creation of `TimingStaffGroup` contexts; in exchange, it allows creating multiple `Staff` contexts that jointly follow the measure defined in their enclosing `TimingStaffGroup`.

### *Locally scaled time signatures*

Use the unscalable `\time` command to establish a measure of the desired length in `Timing`, a.k.a. `TimingStaffGroup`. In this snippet, all staves below `TimingStaffGroup` use a scaled time signature, so any time signature with the desired measure length is as good as any other. If there were an enclosed context that did not use a scaled time signature, the choice of time signature to set in `Timing` would matter in that context.

Use the `\polymetric \time` command to set scalable metric properties in contexts below `Timing`, and use the `\scaleDurations` command to scale both the local meter and the notes to fit the measure.

```
\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \accepts TimingStaffGroup
  }
  \context {
    \StaffGroup
    \name TimingStaffGroup
    \alias StaffGroup
    \consists "Timing_translator"
  }
}

<<
\new TimingStaffGroup <<
  \new Staff {
    \scaleDurations 8/5 {
      \time 6/5 % to set measure length in Timing
      \context Staff \polymetric \time 6/8
      b8 b b b b b
      \time 4/5 % to set measure length in Timing
      \context Staff \polymetric \time 2/4
      b4 b
    }
  }
>>
\new TimingStaffGroup <<
  \new Staff {
    \clef bass
    \time 2/4
    c2 d e f
  }
>>
>>
```



## Chant or psalm notation

This form of notation is used for psalm chant, where verses aren't always of the same length.

```
stemOff = \hide Staff.Stem
```

```
stemOn = \undo \stemOff
```

```
\score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \key g \minor
    \cadenzaOn
    \stemOff a'\breve bes'4 g'4
    \stemOn a'2 \section
    \stemOff a'\breve g'4 a'4
    \stemOn f'2 \section
    \stemOff a'\breve~\markup { \italic flexe }
    \stemOn g'2 \fine
  }
}
```



## Compound time signatures

Odd 20th century time signatures (such as “5/8”) can often be played as complex time signatures (e.g. “3/8 + 2/8”), which combine two or more inequal metrics.

LilyPond can make such music quite easy to read and play, by explicitly printing the time signatures and adapting the automatic beaming behavior.

```
\relative c' {
  \compoundMeter #'((2 8) (3 8))
  \set Timing.beamExceptions = #'()
  \set Timing.beatStructure = 2,3
  c8 d e fis gis
  c8 fis, gis e d
  c8 d e4 gis8
}
```



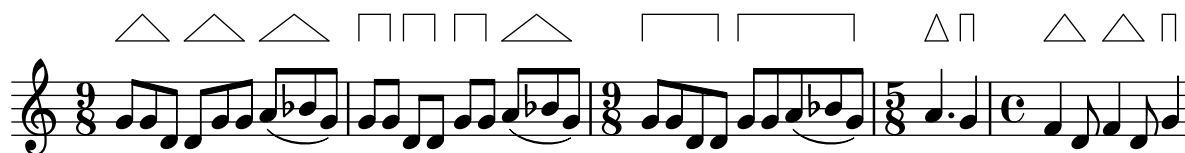
## Conducting signs, measure grouping signs

Context properties control the grouping of beats within a measure: `beatStructure` lists the length of each beat in units of `beatBase`. Default values are established in `scm/time-signature-settings.scm`. These properties may be changed particularly with `\set`.

Alternatively, `\time` optionally accepts a beat structure to use instead of the default. `\time` applies to the Timing context, so it does not reset values of properties that are set in 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`. For the 4/4 measure you have to explicitly set `beatBase` to eighths so that the bar's irregular pattern gets displayed.

```
\score {
  \new Voice \relative c'' {
    \time 9/8
    g8 g d d g g a( bes g) |
    \set Timing.beatStructure = 2,2,2,3
    g8 g d d g g a( bes g) |
    \time 4,5 9/8
    g8 g d d g g a( bes g) |
    \time 5/8
    a4. g4 |
    \time 3,3,2 4/4
    \set Timing.beatBase = #1/8
    f4 d8 f4 d8 g4
  }
  \layout {
    \context {
      \Staff
      \consists "Measure_grouping_engraver"
    }
  }
}
```



## 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), `if-no-beam` (only print a bracket if there is no beam) or `#f` (never print a bracket). The latter is in fact equivalent to omitting the `TupletBracket` object altogether from the printed output.

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



```

\tuplet 3/2 { c8 d e }
\tuplet 3/2 { c4 d e }
}

\new Voice {
  \relative c' {
    \override Score.TextMark.non-musical = ##f
    \textMark "default" \music
    \override TupletBracket.bracket-visibility = #'if-no-beam
    \textMark \markup \typewriter "'if-no-beam" \music
    \override TupletBracket.bracket-visibility = ##t
    \textMark \markup \typewriter "#t" \music
    \override TupletBracket.bracket-visibility = ##f
    \textMark \markup \typewriter "#f" \music
    \omit TupletBracket
    \textMark \markup \typewriter "omit" \music
  }
}

```



## Cow and ride bell example

Two different bells, entered with 'cb' (cow bell) and 'rb' (ride bell).

```

#(define mydrums '((ridebell default #f 3)
                  (cowbell default #f -2)))

\new DrumStaff \with { instrumentName = #"Different Bells" }

\drummode {
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \set DrumStaff.clefPosition = 0.5
  \override DrumStaff.StaffSymbol.line-positions = #'(-2 3)
  \override Staff.BarLine.bar-extent = #'(-1.0 . 1.5)

  \time 2/4
  rb8 8 cb8 16 rb16-> ~ |
  16 8 16 cb8 8 |
}

```



## 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. } #UP
        " = "
        \smaller \general-align #Y #DOWN \note { 8 } #UP
      )
    }
  }
  c1
  c4 c' c,2
}
```



## Engraving ties manually

A single tie may be engraved manually by changing the `staff-position` property (an offset) of the `Tie` grob; if there are multiple ties at the same musical moment, they can be adjusted manually by changing the `tie-configuration` property (a list of offset/direction pairs) of the `TieColumn` object.

The offset indicates the distance from the center of the staff in half-staff spaces, the direction can be either 1 (up) or -1 (down).

Note that LilyPond makes a distinction between exact and inexact values for the offset. If using an exact value (i.e., either an integer or a fraction like  $(/ 4 5)$ ), the value serves as a rough vertical position that gets further tuned by LilyPond to make the tie avoid staff lines. If using an inexact value like a floating point number, it is taken as the precise vertical position without further adjustments.

```
\relative c' {
  <>^"default"
  g'1 ^~ g

  <>^"0"
  \once \override Tie.staff-position = 0
  g1 ^~ g

  <>^"0.0"
  \once \override Tie.staff-position = 0.0
  g1 ^~ g

  <>^"reset"
  \revert Tie.staff-position
  g1 ^~ g
}
```

```

\relative c' {
  \override TextScript.outside-staff-priority = ##f
  \override TextScript.padding = 0

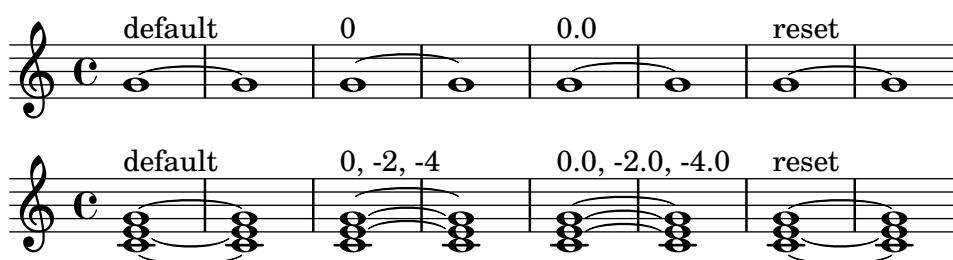
  <>^"default"
  <c e g>1~ <c e g>

  <>^"0, -2, -4"
  \override TieColumn.tie-configuration =
    #'((0 . 1) (-2 . 1) (-4 . 1))
  <c e g>1~ <c e g>

  <>^"0.0, -2.0, -4.0"
  \override TieColumn.tie-configuration =
    #'((0.0 . 1) (-2.0 . 1) (-4.0 . 1))
  <c e g>1~ <c e g>

  <>^"reset"
  \override TieColumn.tie-configuration = ##f
  <c e g>1~ <c e g>
}

```



## Engraving tremolos with floating beams

If a tremolo's total duration is less than a quarter-note, or exactly a half note, or between a half note and a whole note, it is normally typeset with all beams touching the stems. Certain engraving styles typeset some of these beams as centered floating beams that do not touch the stems. The number of floating beams in this type of tremolo is controlled with the `gap-count` property of the `Beam` object, and the size of the gaps between beams and stems is set with the `gap` property.

```

\relative c' {
  \repeat tremolo 8 { a32 f }
  \override Beam.gap-count = #1
  \repeat tremolo 8 { a32 f }
  \override Beam.gap-count = #2
  \repeat tremolo 8 { a32 f }
  \override Beam.gap-count = #3
  \repeat tremolo 8 { a32 f }

  \override Beam.gap-count = #3
  \override Beam.gap = #1.33
  \repeat tremolo 8 { a32 f }
  \override Beam.gap = #1
}

```

```

\repeat tremolo 8 { a32 f }
\override Beam.gap = #0.67
\repeat tremolo 8 { a32 f }
\override Beam.gap = #0.33
\repeat tremolo 8 { a32 f }
}

```



## Entering several tuplets using only one `\tuplet` command

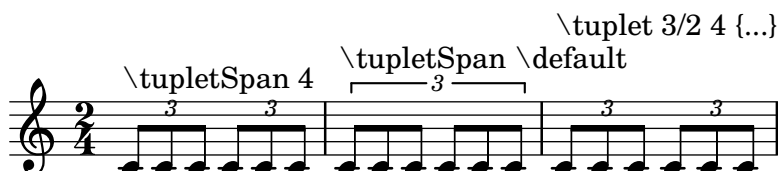
The property `tupletSpannerDuration` sets how long each of the tuplets contained within the brackets after `\tuplet` should last. Many consecutive tuplets can then be placed within a single `\tuplet` expression, thus saving typing.

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

```

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

```



## Forcing rehearsal marks to start from a given letter or number

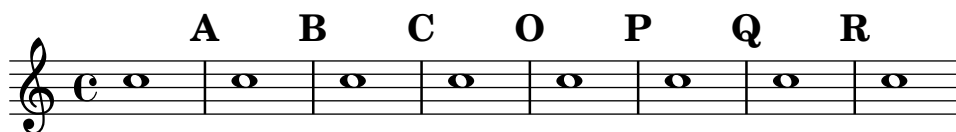
This snippet demonstrates how to obtain automatic ordered rehearsal marks, but from the letter or number desired.

```

\relative c' {
  \override Score.RehearsalMark.Y-offset = #3.5

  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1 \mark #14
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1
}

```



## Generating custom flags

The stencil property of the Flag grob can be set to a custom Scheme function to generate the glyph for the flag.

```
#(define-public (weight-flag grob)
  (let* ((stem-grob (ly:grob-parent grob X))
         (log (- (ly:grob-property stem-grob 'duration-log) 2))
         (is-up? (eqv? (ly:grob-property stem-grob 'direction) UP))
         (yext (if is-up? (cons (* log -0.8) 0) (cons 0 (* log 0.8))))
         (flag-stencil (make-filled-box-stencil '(-0.4 . 0.4) yext))
         (stroke-style (ly:grob-property grob 'stroke-style))
         (stroke-stencil (if (equal? stroke-style "grace")
                              (make-line-stencil 0.2 -0.9 -0.4 0.9 -0.4)
                              empty-stencil)))
    (ly:stencil-add flag-stencil stroke-stencil)))

% Create a flag stencil by looking up the glyph from the font
#(define (inverted-flag grob)
  (let* ((stem-grob (ly:grob-parent grob X))
         (dir (if (eqv? (ly:grob-property stem-grob 'direction) UP) "d" "u"))
         (flag (retrieve-glyph-flag "" dir "" grob))
         (line-thickness (ly:staff-symbol-line-thickness grob))
         (stem-thickness (ly:grob-property stem-grob 'thickness))
         (stem-width (* line-thickness stem-thickness))
         (stroke-style (ly:grob-property grob 'stroke-style))
         (stencil (if (null? stroke-style)
                      flag
                      (add-stroke-glyph flag stem-grob dir stroke-style "")))
         (rotated-flag (ly:stencil-rotate-absolute stencil 180 0 0)))
    (ly:stencil-translate rotated-flag (cons (- (/ stem-width 2)) 0))))

snippetexamplenotes =
{
  \autoBeamOff c'8 d'16 c'32 d'64 \acciaccatura {c'8} d'64
}

{
  \time 1/4
  <>^"Normal flags"
  \snippetexamplenotes

  <>_"Custom flag: inverted"
  \override Flag.stencil = #inverted-flag
  \snippetexamplenotes

  <>^"Custom flag: weight"
```

```

\override Flag.stencil = #weight-flag
\snippetexampnotes

<>_"Revert to normal"
\revert Flag.stencil
\snippetexampnotes
}

```



## Guitar strum rhythms

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

```

\include "predefined-guitar-fretboards.ly"

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

```

C F G C  
 This is my song. I like to sing.

## Heavily customized polymetric time signatures

Though the polymetric time signature shown is not the most essential item here, it has been included to show the beat of this piece (which is the template of a real Balkan song, by the way).

```

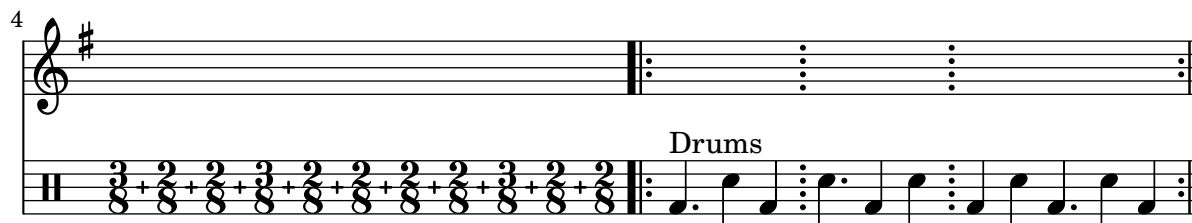
melody = \relative c'' {
  \key g \major
  \compoundMeter #'((3 8) (2 8) (2 8) (3 8) (2 8) (2 8)
                    (2 8) (2 8) (3 8) (2 8) (2 8))
  \set Timing.beamExceptions = #'()
  \set Timing.beatStructure = 3,2,2,3,2,2,2,2,3,2,2
  c8 c c d4 c8 c b c b a4 g fis8 e d c b' c d e4-^ fis8 g \break
  c,4. d4 c4 d4. c4 d c2 d4. e4-^ d4
  c4. d4 c4 d4. c4 d c2 d4. e4-^ d4 \break
}

drum = \new DrumStaff \drummode {
  \repeat volta 2 {
    bd4.^{\markup { Drums }} sn4 bd \bar {"}
    sn4. bd4 sn \bar {"}
    bd sn bd4. sn4 bd
  }
}

\new Staff {
  \melody
  \drum
}

```

This is my song. I like to sing.



## High and low woodblock example

Two Woodblocks, entered with ‘wbh’ (high woodblock) and ‘wbl’ (low woodblock). The length of the bar line has been altered with an `\override` command, otherwise it would be too short. The positions of the two staff lines also have to be explicitly defined.

```
% 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 #f 3)
                  (lowoodblock default #f -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 = #'(-1.0 . 1.5)
  % small correction for the clef:
  \set DrumStaff.clefPosition = 0.5
}
```

```
\new DrumStaff {
  % with this you load your new drum style table
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)

  \woodstaff

  \drummode {
    \time 2/4
    wbh8 16 16 8-> 8 |
    wbl8 16 16-> ~ 16 16 r8 |
  }
}
```



## Making an object invisible using the ‘transparent’ property

Setting the transparent property will cause an object to be printed in “invisible ink”: the object is not printed, but all its other behavior is retained. The object still takes up space, it takes part in collisions, and slurs, ties and beams can be attached to it.



This snippet demonstrates how to connect different voices using ties. Normally, ties only connect two notes in the same voice. By introducing a tie in a different voice, and blanking the first up-stem in that voice, the tie appears to cross voices.

```
\relative {
  \time 2/4
  <<
  {
    \once \hide Stem
    \once \override Stem.length = #8
    b'8 ~ 8\noBeam
    \once \hide Stem
    \once \override Stem.length = #8
    g8 ~ 8\noBeam
  }
  \\\
  {
    b8 g g e
  }
  >>
}

\paper {
  line-width = 40\mm
  ragged-right = ##f
}
```



## Making slurs with complex dash structure

Slurs can be composed of complex dash patterns by setting the `dash-definition` property, which is a list of slur segments, which in turn are lists of parameters setting up the dash behavior of the given segment.

Slur segments are defined in terms of the Bézier parameter  $t$ , which ranges from 0 at the left end of the slur to 1 at the right end of the slur. A slur segment has the form  $(start-t\ stop-t\ dash-fraction\ dash-period)$ . In the segment spanning the range  $start-t$  to  $stop-t$ , the dash pattern is defined by the values of  $dash-fraction$  and  $dash-period$ .  $dash-fraction$  specifies how much of a dash period is black; if set to 1 you get a solid slur segment. The unit for  $dash-period$  is staff spaces.

```
\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)
}
```



## Manually controlling beam positions

Beam positions may be controlled manually, by overriding the positions setting of the Beam grob.

```
\relative c' {
  \time 2/4
  % from upper staff-line (position 2) to center (position 0)
  \override Beam.positions = #'(2 . 0)
  c8 c
  % from center to one above center (position 1)
  \override Beam.positions = #'(0 . 1)
  c8 c
}
```

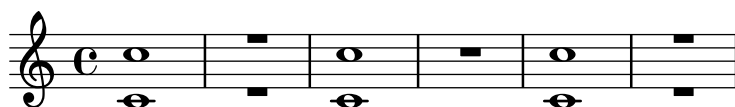


## Merging multi-measure rests in a polyphonic part

Multi-measure rests in a polyphonic staff are placed differently depending on the voice they belong to. They can be printed on the same staff line using the setting below. If you omit the `\once` keyword, the change affects all rests in that follow in the given voice.

```
normalPos = \once \revert MultiMeasureRest.direction
```

```
<<
  { c'1 R c'1 \normalPos R c'1 R } \\
  { c'1 R c'1 \normalPos R c'1 R }
>>
```



## Modifying tuplet bracket length

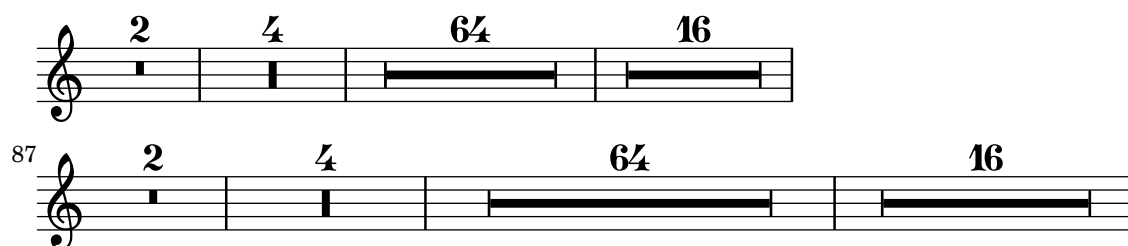
Tuplet brackets can be made to extend horizontally to prefatory matter or the next note. By default, tuplet brackets end at the right edge of the final note of the tuplet; full-length tuplet brackets extend farther to the right, either to cover all the non-rhythmic notation up to the following note, or to cover only the whitespace before the next item of notation, be that a clef, time signature, key signature, or another note. The example shows how to switch tuplets to full length mode and how to modify what material they cover.

```
\new RhythmicStaff {
  % Defaults.
  \time 2/4 \tuplet 3/2 { c4 4 4 }
  \time 4/4 \tuplet 5/4 { 4 1 }
```



```
\override MultiMeasureRest.space-increment = 4
R1*2 R1*4 R1*64 R1*16
}
```

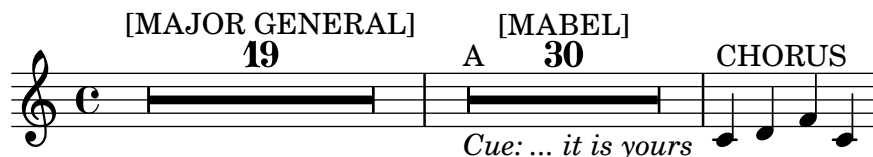
```
\layout {
  ragged-right = ##t
}
```



## Multi-measure rest markup

Markups attached to a multi-measure rest will be centered above or below it. Long markups attached to multi-measure rests do not cause the measure to expand. To expand a multi-measure rest to fit the markup, use an empty chord with an attached markup before the multi-measure rest. Text attached to a spacer rest in this way is left-aligned to the position where the note would be placed in the measure, but if the measure length is determined by the length of the text, the text will appear to be centered.

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



## Non-default tuplet numbers

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

```
\relative c' {
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-tuplet-denominator-text 7)
  \tuplet 3/2 { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-tuplet-fraction-text 12 7)
  \tuplet 3/2 { c4. c4. c4. c4. }
```

```

\once \override TupletNumber.text =
  #(tuplet-number::append-note-wrapper
    (tuplet-number::non-default-tuplet-fraction-text 12 7)
    (ly:make-duration 3 0))
\tuplet 3/2 { c4. c4. c4. c4. }
\once \override TupletNumber.text =
  #(tuplet-number::append-note-wrapper
    tuplet-number::calc-denominator-text
    (ly:make-duration 2 0))
\tuplet 3/2 { c8 c8 c8 c8 c8 c8 }
\once \override TupletNumber.text =
  #(tuplet-number::append-note-wrapper
    tuplet-number::calc-fraction-text
    (ly:make-duration 2 0))
\tuplet 3/2 { c8 c8 c8 c8 c8 c8 }
\once \override TupletNumber.text =
  #(tuplet-number::fraction-with-notes
    (ly:make-duration 2 1) (ly:make-duration 3 0))
\tuplet 3/2 { c4. c4. c4. c4. }
\once \override TupletNumber.text =
  #(tuplet-number::non-default-fraction-with-notes 12
    (ly:make-duration 3 0) 4 (ly:make-duration 2 0))
\tuplet 3/2 { c4. c4. c4. c4. }
}

```



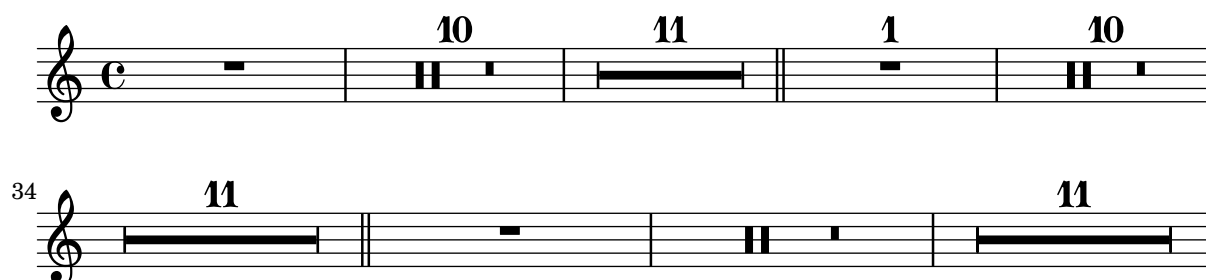
## Numbering single measure rests

Multi-measure rests show their length by a number except for single measures. This can be changed by setting `restNumberThreshold`.

```

{
  \compressEmptyMeasures
  R1 R1*10 R1*11 \bar "||"
  \set restNumberThreshold = 0
  R1 R1*10 R1*11 \bar "||"
  \set restNumberThreshold = 10
  R1 R1*10 R1*11
}

```



## 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 auto-beaming is turned off for the entire staff. Use this at a spot in your score where no beam generated by the auto-beamer is still active.

Internally, `\partCombine` works with four voices – up-stem single, down-stem single, combined, and solo. In order to use `\autoBeamOff` to stop all auto-beaming when used with `\partCombine`, it is necessary to use *four* calls to `\autoBeamOff`.

```
{
% \set Staff.autoBeaming = ##f % turns off all auto-beaming

\partCombine {
  \autoBeamOff % applies to split up-stems
  \repeat unfold 4 a'16
  % \autoBeamOff % applies to combined stems
  \repeat unfold 4 a'8
  \repeat unfold 4 a'16
  % \autoBeamOff % applies to solo
  \repeat unfold 4 a'16
  r4
} {
% \autoBeamOff % applies to split down-stems
\repeat unfold 4 f'8
\repeat unfold 8 f'16 |
r4
\repeat unfold 4 a'16
}
}
```



## Percussion example

A short example taken from Stravinsky's *L'histoire du Soldat*.

```
#(define mydrums '((bassdrum default #f 4)
                   (snare default #f -4)
                   (tambourine default #f 0)))
```

```
U = \stemUp
D = \stemDown
```

```
global = {
  \time 3/8 s4.
  \time 2/4 s2*2
  \time 3/8 s4.
  \time 2/4 s2
}
```

```

drumsA = {
  \context DrumVoice <<
    \global
    \drummode {
      \autoBeamOff
      \D sn8 \U tamb s |
      sn4 \D sn4 |
      \U tamb8 \D sn \U sn16 \D sn \U sn8 |
      \D sn8 \U tamb s |
      \U sn4 s8 \U tamb
    }
  >>
}

drumsB = \drummode {
  s4 bd8 s2*2 s4 bd8 s4 bd8 s
}

\layout {
  indent = 40\mm
  \context {
    \DrumStaff
    drumStyleTable = #(alist->hash-table mydrums)
  }
}

\score {
  \new StaffGroup <<
    \new DrumStaff \with {
      instrumentName = \markup \center-column {
        "Tambourine"
        "et"
        "caisse claire s. timbre" }
    } \drumsA
    \new DrumStaff \with {
      instrumentName = "Grosse Caisse"
    } \drumsB
  >>
}

```

Tambourine  
et  
caisse claire s. timbre

Grosse Caisse

## Permitting line breaks within beamed tuplets

These artificial examples show how both manual and automatic line breaks may be permitted within beamed tuplets that can't be rhythmically split in an exact way.

This feature only works with manually beamed triplets.

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

\relative c' {
  <>^"manually forced line break"
  a8
  \repeat unfold 5 { \tuplet 3/2 { c8[ b g16 a] } }
  \tuplet 3/2 { c8[ b \break g16 a] }
  \repeat unfold 5 { \tuplet 3/2 { c8[ b g16 a] } }
  c8 \bar "||"
}

\relative c' {
  <>^"automatic line break"
  \repeat unfold 28 a16
  \tuplet 11/8 { a16[ b c d e f e d c b a] }
  \repeat unfold 28 a16 \bar "||"
}
```

manually forced line break

automatic line break

The image displays four staves of musical notation. The first two staves are labeled 'manually forced line break' and show beamed triplets of eighth notes. The first staff has a line break after the first triplet, and the second staff has a line break after the second triplet. The last two staves are labeled 'automatic line break' and show beamed triplets of eighth notes. The third staff has a line break after the first triplet, and the fourth staff has a line break after the second triplet. The notation includes various musical symbols such as treble clefs, common time signatures, and beamed notes.

## Positioning grace note beams at the height of normal note beams

When notes are placed on ledger lines, their beams are usually centred on the stave. Grace notes beams are shorter and grace notes on ledger lines may well have beams outside the stave. You can override this beaming for grace notes.

```
\relative c {
```



```
f8[ e]
\grace {
  f8[ e]
  \override Stem.no-stem-extend = ##f
  f8[ e]
  \revert Stem.no-stem-extend
}
f8[ e]
}
```



## 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.

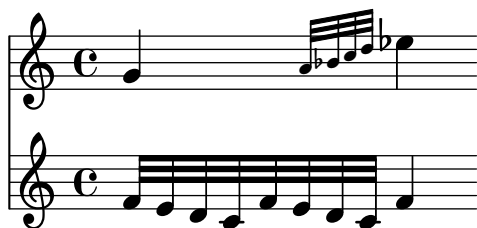
Due to issue #6876 (<https://gitlab.com/lilypond/lilypond/-/issues/6876>), however, accidentals are ignored if this property is set. This snippet gives a workaround to circumvent the problem.

Another unfortunate side effect of this property is that LilyPond does not check whether there is enough horizontal space for grace notes (this is tracked as issue #2630 (<https://gitlab.com/lilypond/lilypond/-/issues/2630>)). You have to make sure that enough space is available, for example, by using `\newSpacingSection` together with a proper value for the `base-shortest-duration` of the `SpacingSpanner` grob.

```
shiftedGrace =
#(define-music-function (offset music) (number? ly:music?)
  #{
    \override NoteHead.X-offset = #(- offset 0.85)
    \override Stem.X-offset = #offset
    \grace { $music }
    \revert NoteHead.X-offset
    \revert Stem.X-offset
  })

\relative c' ' <<
  { g4 \shiftedGrace #-1.3 a32 \shiftedGrace #-0.5 { bes c d } es4 }
  { f,32 e d c f e d c f4 }
>>

\layout {
  \context {
    \Score
    \override SpacingSpanner.strict-grace-spacing = ##t
  }
}
```



## 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.

This snippet shows how positioning of multi-measure rests can be controlled.

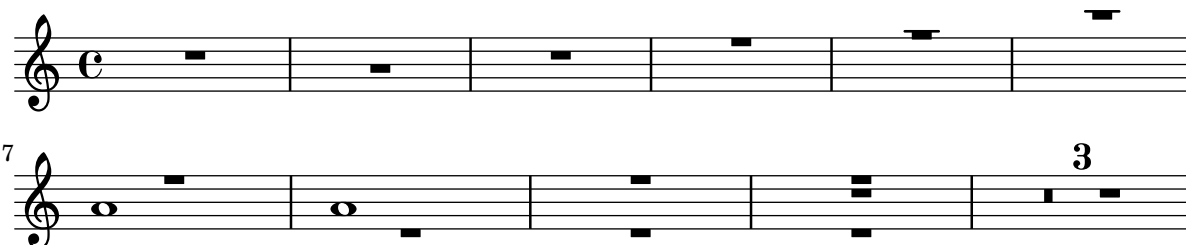
```
\relative c' {
  % Multi-measure rests by default are set under the fourth line.
  R1
  % They can be moved using an override.
  \override MultiMeasureRest.staff-position = -2
  R1
  \override MultiMeasureRest.staff-position = 0
  R1
  \override MultiMeasureRest.staff-position = 2
  R1
  \override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

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

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

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

```
{ R1*3 }
>>
}
```



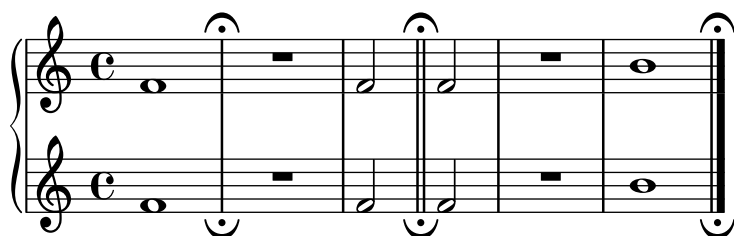
## Positioning opposing fermatas on a bar line

This snippet demonstrates a command that prints fermatas both above and below a bar line. If there would not otherwise be a bar line, it adds a double bar line. Semantically, the command codes a longer-than-normal caesura, which might be considered misuse depending on the situation.

```
twoWayFermata = {
  \once \set Staff.caesuraType = #'((underlying-bar-line . "||"))
  \once \set Staff.caesuraTypeTransform = ##f
  \caesura ^\fermata _\fermata
}
```

```
music = {
  f'1 \twoWayFermata
  R1
  f'2 \twoWayFermata f'2
  R1
  b'1 \twoWayFermata \fine
}
```

```
\new GrandStaff <<
  \new Staff \music
  \new Staff \music
>>
```



## Preventing final mark from removing final tuplet

Due to issue #2362 (<https://gitlab.com/lilypond/lilypond/-/issues/2362>) the addition of a final mark can result in the loss of a final tuplet marking. This can be overcome by setting `TupletBracket.full-length-to-extent` to `#f`.

```
\new Staff {
  \set tupletFullLength = ##t
  \time 1/8
```

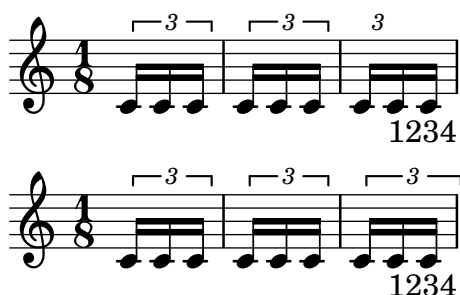
```

\tuplet 3/2 8 { c'16 c' c' c' c' c' c' c' c' }
\tweak direction #DOWN \textEndMark "1234"
}

\new Staff {
  \set tupletFullLength = ##t
  \override TupletBracket.full-length-to-extent = ##f

  \time 1/8
  \tuplet 3/2 8 { c'16 c' c' c' c' c' c' c' c' }
  \tweak direction #DOWN \textEndMark "1234"
}

```



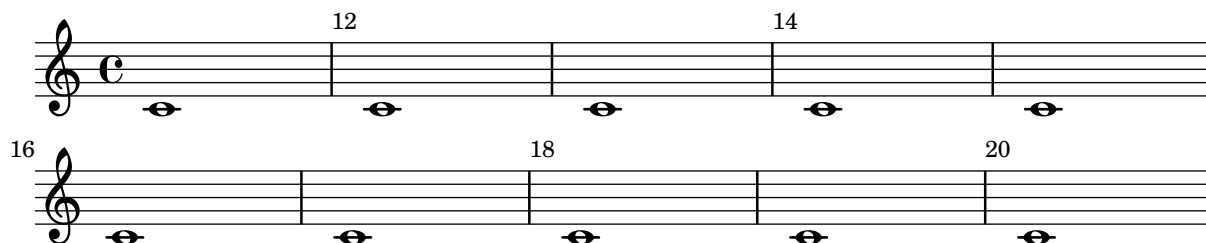
## Printing bar numbers at regular intervals

By setting the `barNumberVisibility` property, bar numbers can be printed at regular intervals. 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
  % 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 for broken measures

By default, a bar number of a broken measure is not repeated at the beginning of the new line. Use `first-bar-number-invisible-save-broken-bars` for `barNumberVisibility` to get a parenthesized BarNumber there.

```

\layout {
  \context {
    \Score
    barNumberVisibility = #first-bar-number-invisible-save-broken-bars
  }
}

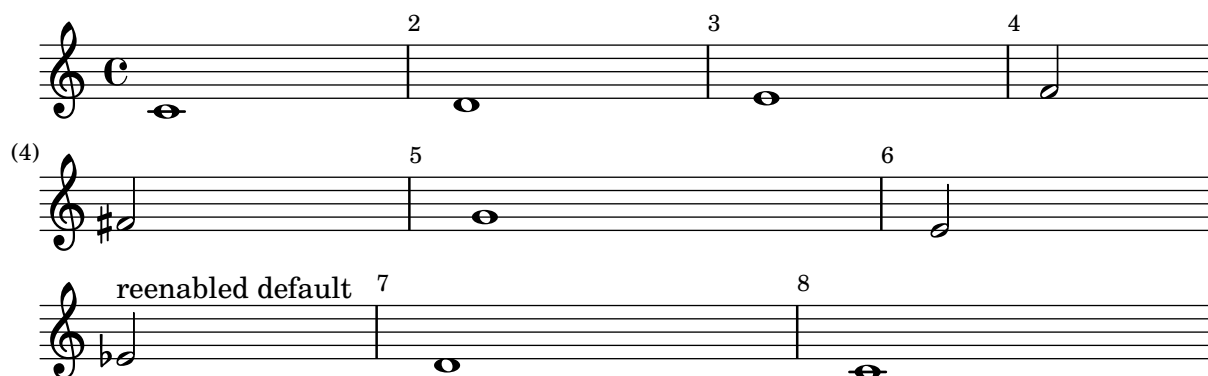
```

```

\override BarNumber.break-visibility = ##(#f #t #t)
}
}

\relative c' {
  c1 | d | e | f2 \break
  fis2 | g1 | e2 \break
  <>^"reenabled default"
  % back to default -
  % \unset Score.barNumberVisibility would do so as well
  \set Score.barNumberVisibility =
    #first-bar-number-invisible-and-no-parenthesized-bar-numbers
  es2 | d1 | c
}

```



## Printing bar numbers inside boxes or circles

Bar numbers can also be printed inside boxes or circles.

```

\relative c' {
  % Center bar numbers except at the beginning of a staff.
  \override Score.BarNumber.self-alignment-X =
    #(break-alignment-list CENTER CENTER 0.3)

  % Prevent bar numbers at the end of a line and permit them elsewhere.
  \override Score.BarNumber.break-visibility = #end-of-line-invisible

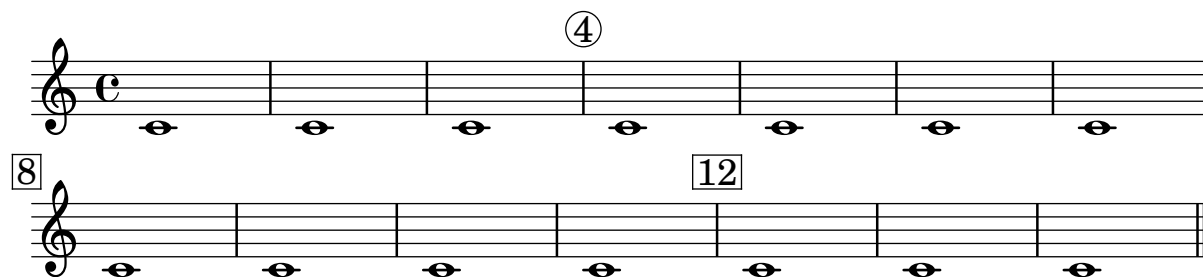
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 4)

  % Increase the size of the bar number by 2.
  \override Score.BarNumber.font-size = 2

  % Draw a circle round the following bar number(s).
  \override Score.BarNumber.stencil
    = #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
  \repeat unfold 7 { c1 } \break

  % 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 7 { c1 } \bar "|"
}

```



## Printing bar numbers using modulo-bar-number-visible

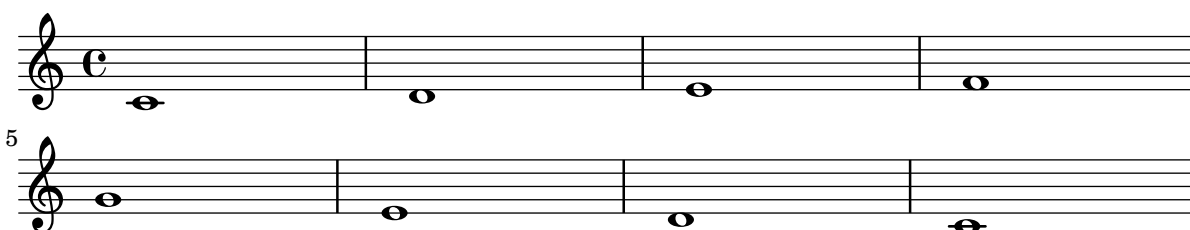
If the remainder of the division of the current bar number by the first argument of `modulo-bar-number-visible` equals its second argument, print a bar number.

This is useful to print the bar number at certain distances. Some examples:

- `(modulo-bar-number-visible 3 2)` → prints 2, 5, 8, ...
- `(modulo-bar-number-visible 4 2)` → prints 2, 6, 10, ...
- `(modulo-bar-number-visible 2 1)` → prints 3, 5, 7, ...
- `(modulo-bar-number-visible 5 0)` → prints 5, 10, 15, ...

```
\layout {
  \context {
    \Score
    \override BarNumber.break-visibility = ##(f #t #t)
    barNumberVisibility = #(modulo-bar-number-visible 5 0)
  }
}
```

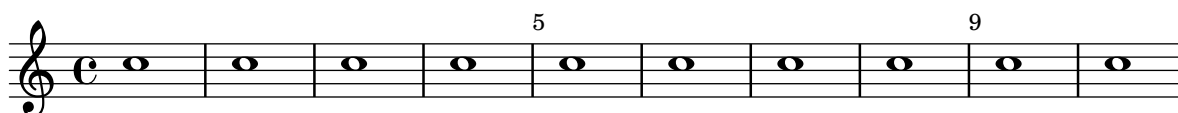
```
\relative c' {
  c1 | d | e | f \break
  g1 | e | d | c
}
```

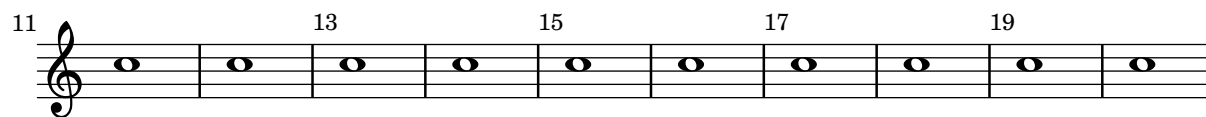


## Printing bar numbers with changing regular intervals

Using the `set-bar-number-visibility` context function, bar number intervals can be changed.

```
\relative c' {
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \context Score \applyContext #(set-bar-number-visibility 4)
  \repeat unfold 10 c'1
  \context Score \applyContext #(set-bar-number-visibility 2)
  \repeat unfold 10 c
}
```





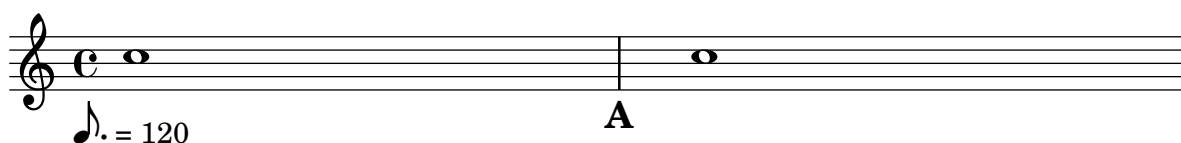
## Printing metronome and rehearsal marks below the staff

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

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

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

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



## Printing music with different time signatures

In the following snippet, two parts have a completely different time signature, yet remain synchronized.

The bar lines can no longer be printed at the `Score` level; to allow independent bar lines in each part, the `Default_barline_engraver` and `Timing_translator` are moved from the `Score` context to the `Staff` context.

If bar numbers are required, the `Bar_number_engraver` should also be moved, since it relies on properties set by the `Timing_translator`; a `\with` block can be used to add bar numbers to the relevant staff.

```
global = {
  \time 3/4 s2.*3 \break
  s2.*3
}

\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Bar_number_engraver"
    \override SpacingSpanner.uniform-stretching = ##t
    \override SpacingSpanner.strict-note-spacing = ##t
    \proportionalNotationDuration = #1/64
  }
```

```

}
\context {
  \Staff
  \consists "Timing_translator"
}
\context {
  \Voice
  \remove "Forbid_line_break_engraver"
  \tupletFullLength = ##t
}
}

Bassklarinette = \new Staff \with {
  \consists "Bar_number_engraver"
  \barNumberVisibility = #(every-nth-bar-number-visible 2)
  \override BarNumber.break-visibility = #end-of-line-invisible
} <<
\global
{
  \clef treble
  \time 3/8 d''4. |
  \time 3/4 r8 des''2( c''8) |
  \time 7/8 r4. ees''2 ~ |
  \time 2/4 \tupletUp \tuplet 3/2 { ees''4 r4 d''4 ~ } |
  \time 3/8 \tupletUp \tuplet 4/3 { d''4 r4 } |
  \time 2/4 e''2 |
  \time 3/8 es''4. |
  \time 3/4 r8 d''2 r8 |
}
>>

Perkussion = \new StaffGroup <<
  \new Staff <<
    \global
    {
      \clef percussion
      \time 3/4 r4 c'2 ~ |
      c'2. |
      R2. |
      r2 g'4 ~ |
      g'2. ~ |
      g'2. |
    }
  >>
  \new Staff <<
    \global {
      \clef percussion
      \time 3/4 R2. |
      g'2. ~ |
      g'2. |
      r4 g'2 ~ |
      g'2 r4 |
    }
  >>

```



```

      g'2. |
    }
  >>
>>

\score {
  <<
    \Bassklarinette
    \Perkussion
  >>
}

```

The musical score consists of three systems. Each system has a treble clef staff and a bass clef staff. The first system starts with a treble clef staff in 3/8 time, followed by a 2/4 time signature, and then a 3/8 time signature. The bass clef staff is in 3/4 time. The second system is marked with a (4) and shows a treble clef staff in 3/8 time, followed by a 2/4 time signature, and then a 3/8 time signature. The bass clef staff is in 3/4 time. The third system is marked with an 8 and shows a treble clef staff in 3/4 time, followed by a 3/8 time signature, and then a 3/4 time signature. The bass clef staff is in 3/4 time.

## Printing the bar number for the first measure

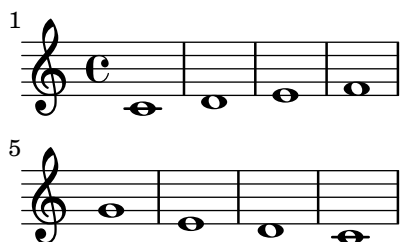
By default, the first bar number in a score is suppressed if it is less than or equal to 1. This can be changed by setting the `barNumberVisibility` context property to value `all-bar-numbers-visible`.

```

\paper {
  line-width = 50\mm
}

```

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



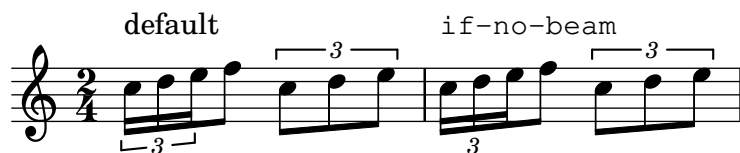
## Printing tuplet brackets on the note head side

Whichever option you choose for controlling the tuplet bracket visibility, it will show or hide the tuplet bracket irrespectively of tuplet bracket placement (stem side or note head side). However, when placing the tuplet bracket on the note head side some authors recommend always printing the tuplet bracket. The option `visible-over-note-heads` can be used to achieve this.

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

```
\new Voice {
  \relative c' {
    \override TextScript.staff-padding = #2.5

    \time 2/4
    \override TupletBracket.visible-over-note-heads = ##t
    \override Score.TextMark.non-musical = ##f
    <>\markup "default" \music
    \override TupletBracket.bracket-visibility = #'if-no-beam
    <>\markup \typewriter "if-no-beam" \music
  }
}
```



## Redefining grace note global defaults

The global defaults for grace notes are stored in the following identifiers.

```
startGraceMusic
stopGraceMusic
startAcciaccaturaMusic
stopAcciaccaturaMusic
startAppoggiaturaMusic
stopAppoggiaturaMusic
```

They are defined in file `ly/grace-init.ly`. By redefining them other effects may be obtained.

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

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

\relative c'' {
  \acciaccatura d8 c1
}
```

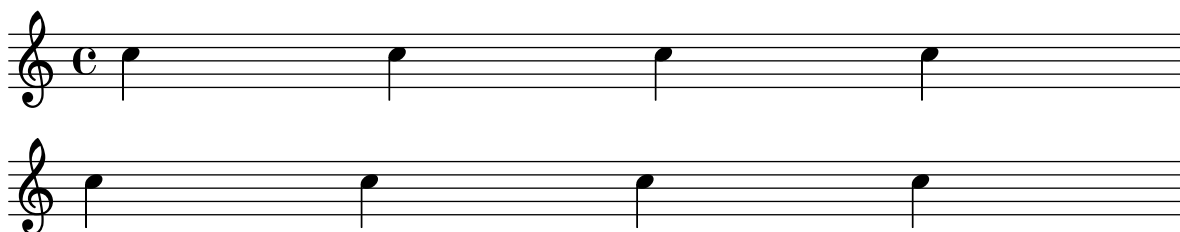


## Removing bar numbers from a score

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

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

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



## Rest styles

Rests may be used in various styles.

```
restsA = {
  r\maxima r\longa r\breve r1 r2 r4 r8 r16 s32
  s64 s128 s256 s512 s1024 s1024
}
```

```

}
restsB = {
  r\maxima r\longa r\breve r1 r2 r4 r8 r16 r32
  r64 r128 r256 r512 r1024 s1024
}

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

  \override Staff.Rest.style = #'mensural
  <>^\markup \typewriter { mensural } \restsA \bar "" \break

  \override Staff.Rest.style = #'neomensural
  <>^\markup \typewriter { neomensural } \restsA \bar "" \break

  \override Staff.Rest.style = #'classical
  <>^\markup \typewriter { classical } \restsB \bar "" \break

  \override Staff.Rest.style = #'z
  <>^\markup \typewriter { z-style } \restsB \bar "" \break

  \override Staff.Rest.style = #'default
  <>^\markup \typewriter { default } \restsB \bar "" \break
}

```

The image displays five musical staves, each illustrating a different rest style. The staves are labeled from top to bottom: mensural, neomensural, classical, z-style, and default. Each staff shows a sequence of rests of increasing duration, with the final rests beamed together. The mensural style uses simple horizontal lines. The neomensural style uses vertical stems with flags. The classical style uses vertical stems with beams. The z-style uses vertical stems with a 'z' symbol. The default style uses vertical stems with a 'z' symbol and a final beam.

## Reverting default beam endings

To typeset beams grouped 3-4-3-2 in 12/8 it is necessary first to override the default beam endings in 12/8, and then to set up the new beaming endings:

```

\relative c'' {
  \time 12/8

```

```

% Default beaming
a8 a a a a a a a a a a

% Set new values for beam endings
\set Score.beatStructure = 3,4,3,2
a8 a a a a a a a a a a
}

```



## Rhythmic slashes

In “simple” lead-sheets, sometimes no actual notes are written. Instead, only “rhythmic patterns” and chords above the measures are notated to represent the structure of a song. Such a feature can be useful while creating or transcribing the structure of a song, or when sharing lead sheets with guitarists or jazz musicians.

```

startPat = {
  \improvisationOn
  \omit Stem
}
stopPat = {
  \improvisationOff
  \undo \omit Stem
}

\new Voice \with {
  \consists Pitch_squash_engraver
} {
  c'4 d' e' f' |
  \startPat
  4 4 4 4 |
  \stopPat
  f'4 e' d' c'
}

```



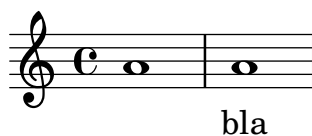
## Skips in lyric mode

The ‘s’ syntax for skips is only available in note mode and chord mode. In other situations, for example, when entering lyrics, using the `\skip` command is recommended.

```

<<
  \relative c'' { a1 | a }
  \new Lyrics \lyricmode { \skip1 bla1 }
>>

```



## Skips in lyric mode (2)

Although ‘s’ skips cannot be used in `\lyricmode` (it is taken to be a literal “s”, not a space), double quotes (“”) or underscores (\_) are available.

```
<<
\relative c' { a4 b c d }
\new Lyrics \lyricmode { a4 "" _ gap }
>>
```



## Stemlets

In some notational conventions beams are allowed to extend over rests. Depending on preference, these beams may drop ‘stemlets’ to help the eye appreciate the rhythm better, and in some modern music the rest itself is omitted and only the stemlet remains.

This snippet shows a progression from traditional notation, to beams over the rest, to stemlets over the rest, to stemlets alone. Stemlets are generated by overriding the `stemlet-length` property of `Stem`, and rests are hidden by using `\hide`.

Some `\markup` elements are included in the source to highlight the different notations.

```
\paper {
  ragged-right = ##f
}

{
  c'16^\markup { traditional } d' r f'
  g'16[^\markup \column { "beams" "over rests" } f' r d']

  % N.B. use Score.Stem to set for the whole score.
  \override Staff.Stem.stemlet-length = #0.75

  c'16[^\markup \column { "stemlets" "over rests" } d' r f']
  g'16[^\markup \column { "stemlets" "and no rests" } f'
  \once \hide Rest
  r16 d']
}
```



## 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
}
```



## Subdividing beams

The beams of consecutive 16th (or shorter) notes are, by default, not subdivided. That is, the beams of more than two stems stretch over the entire group of notes without a break. This behavior can be modified to subdivide the beams into sub-groups by setting the property `subdivideBeams` to `#t`. When set, beams are subdivided at (rhythmic) intervals to match the metric value of the subdivision.

Using the properties `beamMinimumSubdivision` and `beamMaximumSubdivision` it is possible to configure the limits of automatic beam subdivision, namely the minimum and maximum rhythmic lengths at which beamlets are removed. The default values are 0 for the former and `+inf.0` for the latter, making LilyPond subdivide beams as much as possible.

There are two special cases to consider.

- If the numerator of `beamMaximumSubdivision` is not a power of 2, the rhythmic lengths considered for subdivision are `beamMaximumSubdivision` divided by powers of 2 that stay greater than or equal to `beamMinimumSubdivision`.
- If `beamMaximumSubdivision` is smaller than `beamMinimumSubdivision`, the depth of beam subdivisions is limited by `beamMaximumSubdivision`, but not the frequency and rhythmic intervals, therefore possibly deviating from the correct, expected metric value.

If `respectIncompleteBeams` is set to `#t`, incomplete subdivisions with more than two stems are treated as an ‘extension’ of the previous subdivision group, i.e., the length of the previous subdivision group gets extended to also cover the incomplete subdivision. If set to `#f` (which is the default), a new subdivision group gets started instead.

```
\relative c' {
  \time 1/4

  <>^"default"
  c32 c c c c c c c

  <>^"with subdivision"
  \set subdivideBeams = ##t
  c32 c c c c c c c

  <>^"min 1/8"
  \once \set beamMinimumSubdivision = #1/8
  c32 c c c c c c c

  <>^"max 1/16"
  \once \set beamMaximumSubdivision = #1/16
  c32 c c c c c c c

  <>^"max 3/8"
  \once \set beamMaximumSubdivision = #3/8
  \repeat unfold 16 c64
```

```

<>~"min 1/32, max 1/64"
% Set maximum beam subdivision interval to 1/64 to limit
% subdivision depth, despite not being metrically correct.
\once \set beamMinimumSubdivision = #1/32
\once \set beamMaximumSubdivision = #1/64
\repeat unfold 32 c128
\break

<>~"beams with incomplete subdivisions"
c32 c c c c c c c r32
c32 c c c c c r16.

<>~\markup { "the same with"
      \typewriter { "respectIncomplete=#t" } }
\set respectIncompleteBeams = ##t
% The incomplete subgroup extends the completed subgroup.
c32 c c c c c c c r32
% No visual change since we have only two stems in the
% incomplete subgroup.
c32 c c c c c r16.
}

```

The image displays three staves of musical notation in 4/4 time, illustrating different beam subdivision settings. The first staff shows four measures: 'default' (8 eighth notes), 'with subdivision' (8 eighth notes with a beam), 'min 1/8' (8 eighth notes with a beam), and 'max 1/16' (8 eighth notes with a beam). The second staff shows two measures: 'max 3/8' (a half note) and 'min 1/32, max 1/64' (a half note with a beam). The third staff shows four measures: 'beams with incomplete subdivisions' (8 eighth notes with a beam), 'the same with respectIncomplete=#t' (8 eighth notes with a beam), and two more measures showing the same notation with a different beam setting.

## Tam-tam example

A tam-tam example, entered with 'tt'.

```

#(define mydrums '((tamtam default #f 0)))

\new DrumStaff \with { instrumentName = #"Tamtam" }

\drummode {
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \override Staff.StaffSymbol.line-positions = #'( 0 )
  \override Staff.BarLine.bar-extent = #'(-1.5 . 1.5)

  tt 1 \pp \laissezVibrer
}

```

Tamtam *pp*



## Tambourine example

A tambourine example, entered with ‘tamb’.

```
#(define mydrums '((tambourine default #f 0)))

\new DrumStaff \with { instrumentName = #"Tambourine" }

\drummode {
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \override Staff.StaffSymbol.line-positions = #'( 0 )
  \override Staff.BarLine.bar-extent = #'(-1.5 . 1.5)

  \time 6/8
  tamb8. 16 8 8 8 8 |
  tamb4. 8 8 8 |
  % The trick with the scaled duration and the shorter rest
  % is necessary for the correct ending of the trill-span!
  tamb2.*5/6 \startTrillSpan s8 \stopTrillSpan |
}
```



## Three-sided box

This example shows how to add a markup command to get a three-sided box around some text (or other markup).

```
% New command to add a three-sided box, with sides north, west, and south.
% Based on the `box-stencil` command defined in `scm/stencil.scm`.
% Note that ";;" is used to comment a line in Scheme.
#(define-public (NWS-box-stencil stencil thickness padding)
  "Add a box around STENCIL, producing a new stencil."
  (let* ((x-ext (interval-widen (ly:stencil-extent stencil X) padding))
        (y-ext (interval-widen (ly:stencil-extent stencil Y) padding))
        (y-rule (make-filled-box-stencil (cons 0 thickness) y-ext))
        (x-rule (make-filled-box-stencil
                  (interval-widen x-ext thickness) (cons 0 thickness))))
    ;; (set! stencil (ly:stencil-combine-at-edge stencil X 1 y-rule padding))
    (set! stencil (ly:stencil-combine-at-edge stencil X LEFT y-rule padding))
    (set! stencil (ly:stencil-combine-at-edge stencil Y UP x-rule 0.0))
    (set! stencil (ly:stencil-combine-at-edge stencil Y DOWN x-rule 0.0))
    stencil))

% The corresponding markup command, based on the `\\box` command defined
% in `scm/define-markup-commands.scm`.
#(define-markup-command (NWS-box layout props arg) (markup?)
  #:properties ((thickness 0.1) (font-size 0) (box-padding 0.2))
  "Draw a box round ARG.
```

Look at THICKNESS, BOX-PADDING, and FONT-SIZE properties to determine line thickness and padding around the markup."

```
(let ((pad (* (magstep font-size) box-padding))
```

```
(m (interpret-markup layout props arg)))
(NWS-box-stencil m thickness pad)))
```

```
\relative c' {
  c1~\markup { \NWS-box ABCD }
  c1~\markup { \NWS-box \note {4} #1.0 }
}
```



## Time signature in brackets

The time signature can be enclosed within brackets.

```
\relative c' {
  \override Staff.TimeSignature.stencil = #(lambda (grob)
    (bracketify-stencil (ly:time-signature::print grob) Y 0.1 0.2 0.1))
  \time 2/4
  a4 b8 c
}
```



## Time signature in parentheses

The time signature can be enclosed within parentheses.

```
\relative c' {
  \override Staff.TimeSignature.stencil = #(lambda (grob)
    (parenthesize-stencil (ly:time-signature::print grob) 0.1 0.4 0.4 0.1))
  \time 2/4
  a4 b8 c
}
```



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

Sometimes, a time signature should not print the whole fraction (for example, 7/4), but only the numerator (digit 7 in this case). This can be easily done by using `\override Staff.TimeSignature.style = #'single-number` to change the style permanently. By using `\revert Staff.TimeSignature.style`, this setting can be reversed. To apply the single-number 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-number
\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-number style only for the next time signature
\once \override Staff.TimeSignature.style = #'single-number
\time 5/4
c4 c c c c
\time 2/4
c4 c
}

```



## Tweaking grace layout within music

The appearance of grace expressions can be changed by using the functions `add-grace-property` and `remove-grace-property`.

The following example undefines the `direction` property of Stem grobs 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
    }
  }
}

```



## User-defined time signatures

New time signature styles can be defined. The time signature in the second measure is printed upside down in both staves.

```

#(add-simple-time-signature-style 'topsy-turvy
  (lambda (fraction)
    (make-rotate-markup 180 (make-compound-meter-markup fraction))))

```

<<

```

\new Staff {
  \time 3/4 f'2.
  \override Score.TimeSignature.style = #'topsy-turvy
  \time 3/4 R2. \bar "|."
}
\new Staff {
  R2. e''
}
>>

```



## Using alternative flag styles

Alternative shapes for flags on eighth and shorter notes can be displayed by overriding the `stencil` property of `Flag`. LilyPond provides the following functions: `modern-straight-flag`, `old-straight-flag`, and `flat-flag`. Use `\revert` to restore the default shape.

To get stacked (i.e., vertically more compact) flags, call the command `\flagStyleStacked`, which can be reset with `\flagStyleDefault`.

Overriding the `Flag` stencil does not change how flag elements are positioned vertically. This is especially noticeable for flat flags: LilyPond doesn't dynamically adjust the vertical gaps between flag elements in the same way as it does for beams. A possible solution to harmonize the appearance is to replace flat flags with half beams, as shown in the second staff; however, this can't be done automatically. In the code of this snippet, such half beams are entered with `@` as a prefix, for example `@c8`.

Be aware that half beams are *not* `Flag` grobs. This means in particular that modifying `Flag` properties won't have any effect on them (you have to use `Beam` properties instead), and properties for their associated `Stem` grob will also behave beam-like.

```

"@ =
#(define-music-function (music) (ly:music?)
  #{ \set stemLeftBeamCount = 0 $music [] #})

testnotes = {
  \autoBeamOff
  c8 d16 e''32 f64 \acciaccatura { g,,,8 } a128 b
}

\relative c' {
  \override TextScript.staff-padding = 6
  \time 1/4
  <>^"default" \testnotes
  \override Flag.stencil = #modern-straight-flag
  <>_"modern straight" \testnotes
  \override Flag.stencil = #old-straight-flag
  <>^"old straight" \testnotes
  \override Flag.stencil = #flat-flag
}

```

```

    <>_"flat" \testnotes
\revert Flag.stencil

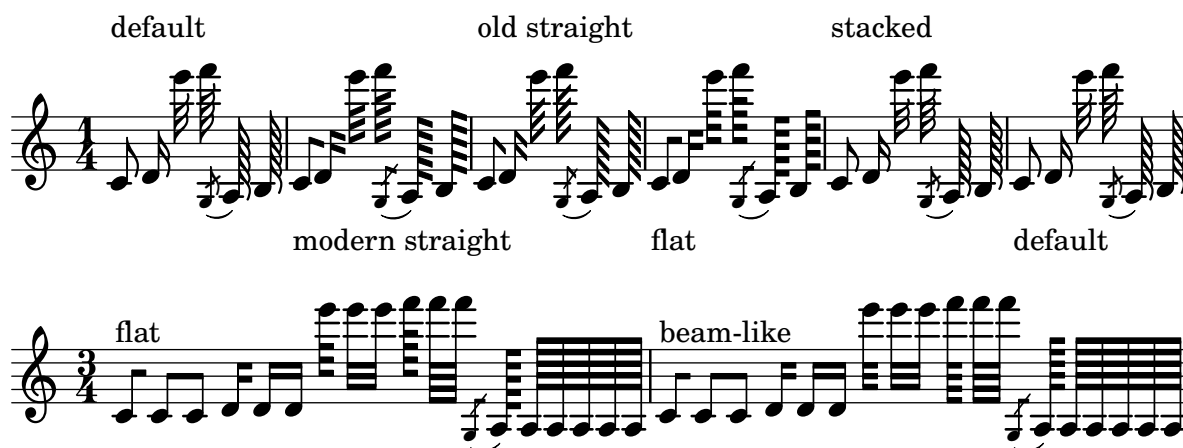
\flagStyleStacked
    <>^"stacked" \testnotes
\flagStyleDefault
    <>_"default" \testnotes
}

\relative c' {
    \time 3/4
    \override Flag.stencil = #flat-flag

    <>^"flat" c8 c[ c] d16 d[ d] e''32 e[ e] f64 f[ f]
        \acciaccatura { g,,,8 } a128 a[ a a a a]
    <>^"beam-like" @c8 c[ c] @d16 d[ d] @e''32 e[ e] @f64 f[ f]
        \acciaccatura { g,,,8 } @a128 a[ a a a a]
}

\layout {
    indent = 0
    \context {
        \Score
        \override NonMusicalPaperColumn.line-break-permission = ##f
    }
}

```



## 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)
}

```



## 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
}
```



### 3 Expressive marks

See also Section “Expressive marks” in *Notation Reference*.

#### Adding parentheses around an expressive mark or chordal note

The `\parenthesize` function is a special tweak that encloses objects in parentheses. The associated grob is `Parentheses`.

```
\relative c' {
  c2-\parenthesize ->
  \override Parentheses.padding = #0.1
  \override Parentheses.font-size = #-4
  <d \parenthesize f a>2
}
```



#### 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 |
}

```



## Adjusting slur positions vertically

Using `\override Slur.positions` it is possible to set the vertical position of the start and end points of a slur to absolute values (or rather, forcing LilyPond's slur algorithm to consider these values as desired). In many cases, this means a lot of trial and error until good values are found. You probably have tried the `\offset` command next just to find out that it doesn't work for slurs, emitting a warning instead.

The code in this snippet allows you to tweak the vertical start and end positions by specifying *relative* changes, similar to `\offset`.

```

Syntax: \offsetPositions #'(dy1 . dy2)

offsetPositions =
#(define-music-function (offsets) (number-pair?)
  #{
    \once \override Slur.control-points =
      #(\lambda (grob)
        (match-let (((_ . y1) _ _ (_ . y2))
                    (ly:slur::calc-control-points grob))
          ((off1 . off2) offsets))
        (set! (ly:grob-property grob 'positions)
              (cons (+ y1 off1) (+ y2 off2)))
          (ly:slur::calc-control-points grob)))
      #})

\relative c' {
  c4(^"default" c, d2)
  \offsetPositions #'(0 . 1)
  c'4(^"(0 . 1)" c, d2)
  \offsetPositions #'(0 . 2)
  c'4(^"(0 . 2)" c, d2)
  \bar "||"
  g4(^"default" a d'2)
  \offsetPositions #'(1 . 0)
  g,,4(^"(1 . 0)" a d'2)
  \offsetPositions #'(2 . 0)
  g,,4(^"(2 . 0)" a d'2)
}

```





## 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
}
```



## Aligning the ends of hairpins to NoteColumn directions

The ends of hairpins may be aligned to the `LEFT`, `CENTER`, or `RIGHT` of `NoteColumn` grobs by overriding the property `endpoint-alignments`, which is a pair of numbers representing the left and right ends of the hairpin. `endpoint-alignments` are expected to be directions (either -1, 0 or 1). Other values will be transformed with a warning. The right end of a hairpin terminating at a rest is not affected, always ending at the left edge of the rest.

```
{
  c'2\< <c' d'\>\! |
  \override Hairpin.endpoint-alignments = #'(1 . -1)
  c'2\< <c' d'\>\! |
  \override Hairpin.endpoint-alignments = #'(,LEFT . ,CENTER)
  c'2\< <c' d'\>\! |
}
```



## Alternative breve notes

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

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



## Asymmetric slurs

Slurs can be made asymmetric to match an asymmetric pattern of notes better.

```
slurNotes = { d,8( a' d f a f' d, a) }
```

```
\relative c' {
  \stemDown
  \slurUp
  \slurNotes
  \once \override Slur.eccentricity = #3.0
  \slurNotes
}
```



## Breathing signs

Breathing signs are available in different tastes: commas (default), ticks, vees and “railroad tracks” (caesura).

```
\new Staff \relative c'' {
  \key es \major
  \time 3/4
  % this bar contains no \breathe
  << { g4 as g } \ { es4 bes es } >> |
  % Modern notation:
  % by default, \breathe uses the rcomma, just as if saying:
  % \override BreathingSign.text =
  %   #(make-musicglyph-markup "scripts.rcomma")
  << { g4 as g } \ { es4 \breathe bes es } >> |

  % rvarcomma and lvarcomma are variations of the default rcomma
  % and lcomma
  % N.B.: must use Staff context here, since we start a Voice below
  \override Staff.BreathingSign.text =
    \markup { \musicglyph "scripts.rvarcomma" }
  << { g4 as g } \ { es4 \breathe bes es } >> |

  % raltcomma and laltcomma are alternative variations of the
  % default rcomma and lcomma
  \override Staff.BreathingSign.text =
    \markup { \musicglyph "scripts.raltcomma" }
  << { g4 as g } \ { es4 \breathe bes es } >> |

  % vee
  \override BreathingSign.text =
    \markup { \musicglyph "scripts.uupbow" }
  es8[ d es f g] \breathe f |

  % caesura
```

```
\override BreathingSign.text =
  \markup { \musicglyph "scripts.caesura.curved" }
es8[ d] \breathe es[ f g f] |
es2 r4 \bar "||"
}
```



## Broken crescendo hairpin

In order to make parts of a crescendo hairpin invisible, the following method is used: A white rectangle is drawn on top of the respective part of the crescendo hairpin, making it invisible. The rectangle is defined as a text markup.

The markup command with-dimensions tells LilyPond to consider only the bottom edge of the rectangle when spacing it against the hairpin. The property staff-padding prevents the rectangle from fitting between the hairpin and staff.

Make sure the hairpin is in a lower layer than the text markup to draw the rectangle over the hairpin.

```
\relative c' {
  <<
  {
    \dynamicUp
    r2 r16 c'8.\pp r4
  }
  \\\
  {
    \override DynamicLineSpanner.layer = #0
    des,2\mf\< ~
    \override TextScript.layer = #2
    \once\override TextScript.staff-padding = #6
    \once\override TextScript.vertical-skylines = #'()
    des16_\markup \with-dimensions #'(2 . 7) #'(0 . 0)
      \with-color #white
      \filled-box #'(2 . 7) #'(0 . 2) #0
    r8. des4 ~ des16->\sff r8.
  }
  >>
}
```



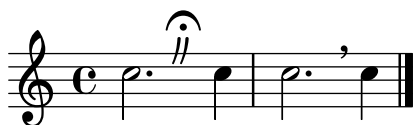
## Caesura (“railtracks”) with fermata

A caesura is sometimes denoted by a double “railtracks” breath mark with a fermata sign positioned above. This snippet shows an optically pleasing combination of railtracks and fermata.

```

\relative c' {
  c2.
  % construct the symbol
  \override BreathingSign.text = \markup {
    \override #'(direction . 1)
    \override #'(baseline-skip . 1.8)
    \dir-column {
      \translate #'(0.155 . 0)
      \center-align \musicglyph "scripts.caesura.curved"
      \center-align \musicglyph "scripts.ufermata"
    }
  }
  \breathe c4
  % set the breath mark back to normal
  \revert BreathingSign.text
  c2. \breathe c4
  \bar "|."
}

```



## Center text below hairpin dynamics

This example provides a function to typeset a hairpin (de)crescendo with some additional text below it, such as “molto” or “poco”. The added text will change the direction according to the direction of the hairpin. The Hairpin is aligned to a DynamicText grob.

The example also illustrates how to modify the way an object is normally printed, using some Scheme code.

```

hairpinWithCenteredText =
#(define-music-function (text) (markup?)
  #{
    \once \override Voice.Hairpin.after-line-breaking =
      #(lambda (grob)
        (let* ((stencil (ly:hairpin::print grob))
              (par-y (ly:grob-parent grob Y))
              (dir (ly:grob-property par-y 'direction))
              (staff-line-thickness
                (ly:output-def-lookup (ly:grob-layout grob)
                                      'line-thickness)))
          (new-stencil
            (ly:stencil-aligned-to
              (ly:stencil-combine-at-edge
                (ly:stencil-aligned-to stencil X CENTER)
                Y dir
                (ly:stencil-aligned-to
                  (grob-interpret-markup
                    grob
                    (make-fontsize-markup
                      (magnification->font-size
                        (+ (ly:staff-symbol-staff-space grob)

```

```

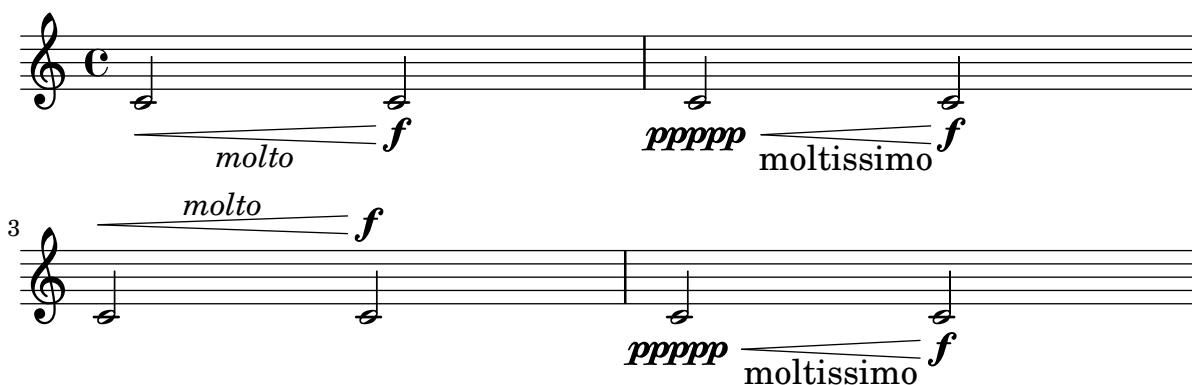
        (/ staff-line-thickness 2)))
      text))
      X CENTER))
      X LEFT))
    (staff-space (ly:output-def-lookup
      (ly:grob-layout grob) 'staff-space))
    (par-x (ly:grob-parent grob X))
    (dyn-text (grob::has-interface par-x
      'dynamic-text-interface))

    (dyn-text-stencil-x-length
      (if dyn-text
        (interval-length
          (ly:stencil-extent
            (ly:grob-property par-x 'stencil) X))
          0))
      (x-shift
        (if dyn-text (- (+ staff-space dyn-text-stencil-x-length)
          (* 0.5 staff-line-thickness))
          0)))
    (ly:grob-set-property! grob 'Y-offset 0)
    (ly:grob-set-property! grob
      'stencil (ly:stencil-translate-axis
        new-stencil
        x-shift X))))
  #})

hairpinMolto = \hairpinWithCenteredText \markup { \italic molto }
hairpinMore = \hairpinWithCenteredText \markup { \larger moltissimo }

\relative c' {
  \hairpinMolto c2\< c\f
  \hairpinMore c2\ppppp\< c\f
  \break
  \hairpinMolto c2^\< c\f
  \hairpinMore c2\ppppp\< c\f
}

```



## Changing text and spanner styles for text dynamics

The text used for *crescendos* and *decrescendos* can be changed by modifying the context properties `crescendoText` and `decrescendoText`.

The style of the spanner line can be changed by modifying the style property of `DynamicTextSpanner`. The default value is dashed-line, and other possible values include line, dotted-line, and none.

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



## Changing the appearance of a slur from solid to dotted or dashed

The appearance of slurs may be changed from solid to dotted or dashed.

```
\relative c' {
  c4( d e c)
  \slurDotted
  c4( d e c)
  \slurSolid
  c4( d e c)
  \slurDashed
  c4( d e c)
  \slurSolid
  c4( d e c)
}
```



## 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
}
```



## Changing the number of augmentation dots per note

The number of augmentation dots on a single note can be overridden by setting the `dot-count` property of the `Dots` grob.

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



## Combining dynamics with markup texts

Some dynamics may involve text indications (such as “*più f*” or “*p subito*”). These can be produced using a `\markup` block; the resulting object behaves like a `TextScript` grob.

See also “Combining dynamics with markup texts (2)”.

```
piuF = \markup { \italic più \dynamic f }
```

```
\markup \with-true-dimensions % work around a cropping issue
\score {
  \relative c'' {
    c2\f c-\piuF
  }
}
```



## Combining dynamics with markup texts (2)

Some dynamics may involve text indications (such as “*più f*” or “*p subito*”). These can be produced using the `make-dynamic-script` Scheme function; the resulting object behaves like a `DynamicText` grob.

See also “Combining dynamics with markup texts”.

```
piuF = #(make-dynamic-script
  #{ \markup { \normal-text \italic più \dynamic f } #})

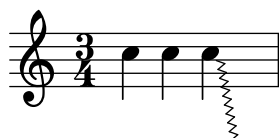
\score {
  \relative c'' {
    c2\f c\piuF
  }
}
```



## 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 "|"
}
```



## Controlling spanner visibility after a line break

The visibility of spanners which end on the first note following a line break is controlled by the after-line-breaking callback `ly:spanner::kill-zero-spanned-time`.

For objects such as glissandos and hairpins, the default behaviour is to hide the spanner after a break; disabling the callback will allow the left-broken span to be shown.

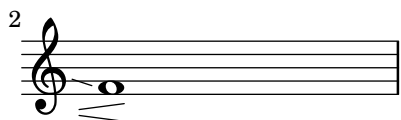
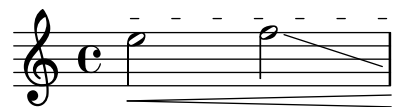
Conversely, spanners which are usually visible, such as text spans, can be hidden by enabling the callback.

```
\paper {
  line-width = 50\mm
}

\relative c' ' {
  \override Hairpin.to-barline = ##f
  \override Glissando.breakable = ##t
  % show hairpin
  \override Hairpin.after-line-breaking = ##t
  % hide text span
  \override TextSpanner.after-line-breaking =
    #ly:spanner::kill-zero-spanned-time
  e2\<\startTextSpan
  % show glissando
  \override Glissando.after-line-breaking = ##t
  f2\glissando
  \break
}
```



```
f,1\!\stopTextSpan
}
```

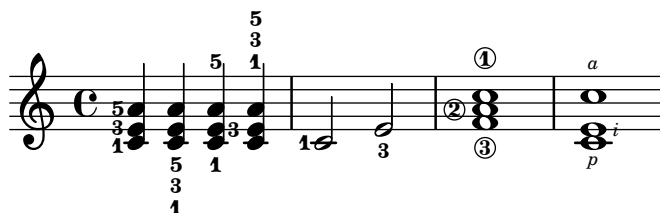


## Controlling the placement of chord fingerings

The placement of fingering numbers can be controlled precisely by using the property `fingeringOrientations`. For fingering orientation to apply, the fingering command must be used within a chord construct (`<...>`), even for single notes. Orientation for string numbers and right-hand fingerings may be controlled in a similar way by using the properties `stringNumberOrientations` and `strokeFingerOrientations`, respectively.

These properties can be set to a list of one to three values. They control whether fingerings may be placed above (if `up` appears in the list), below (if `down` appears), to the left (if `left` appears), or to the right (if `right` appears). Conversely, if a location is not listed, no fingering is placed there. LilyPond takes these constraints and works out the best placement for the fingering of the notes of the following chords. Note that `left` and `right` are mutually exclusive – fingerings may be placed only on one side or the other, not both.

```
\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
  \set stringNumberOrientations = #'(up left down)
  <f\3 a\2 c\1>1
  \set strokeFingerOrientations = #'(down right up)
  <c\rightHandFinger 1 e\rightHandFinger 2 c'\rightHandFinger 4 >
}
```



## 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.

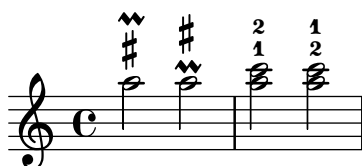
Note that for `Fingering`, `StringNumber`, and `StrokeFinger` grobs, if used within a chord, the vertical order is also determined by the vertical position of the associated note head, which is added to (or, depending on the direction, subtracted from) the grob’s `script-priority` value. This ensures that for fingerings above a chord the lower note is associated with the lower fingering (and vice versa for the other direction); it doesn’t matter whether you input the notes in the chord from top to bottom or from bottom to top.

By default, the least technical scripts are positioned closest to the note head; the rough order is articulation, flageolet, fingering, right-hand fingering, string number, fermata, bowing, and text script.

```
\relative c''' {
  \once \override TextScript.script-priority = -100
  a2^\prall^\markup { \sharp }

  \once \override Script.script-priority = -100
  a2^\prall^\markup { \sharp }

  \set fingeringOrientations = #'(up)
  <c-2 a-1>2
  <a-1 c>\tweak script-priority -100 -2>2
}
```



## Creating “real” parenthesized dynamics

Although the easiest way to add parentheses to a dynamic mark is to use a `\markup` block, this method has a downside: the created objects behave like text markups and not like dynamics.

However, it is possible to create a similar object using the equivalent Scheme code (as described in the Notation Reference), combined with the `make-dynamic-script` function. This way, the markup is regarded as a dynamic and therefore remains compatible with commands such as `\dynamicUp` or `\dynamicDown`.

```
paren =
#(define-event-function (dyn) (ly:event?)
  (make-dynamic-script
    #{ \markup \concat {
      \normal-text \italic \fontsize #2 (
        \pad-x #0.2 #(ly:music-property dyn 'text)
        \normal-text \italic \fontsize #2 )
      }
    })
  #}))
```

```
\relative c' ' {
  c4\paren\f c c \dynamicUp c\paren\p
}
```



## 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 first argument of `\after` (which is a duration) adjusts the horizontal position.

```
\relative c' ' {
  \after 2*2/3 \turn c2( d4) r |
  \after 4 \turn c4.( d8)
  \after 4
  {
    \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
    \once \hideNotes
    cis8\turn \noBeam
  }
  d4.( e8)
}
```



## 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.

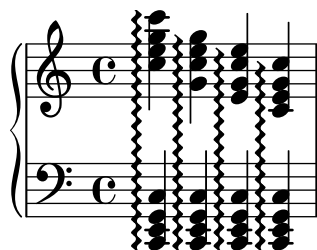
```
\new Staff \with {
  \consists "Span_arpeggio_engraver"
}
\relative c' {
  \set Staff.connectArpeggios = ##t
  <<
    { <e' g>4\arpeggio <d f> <d f>2 }
    \\\
    { <d, f>2\arpeggio <g b>2 }
  >>
}
```



## 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` and its derived siblings (`PianoStaff`, `ChoirStaff`, and `StaffGroup`) if the `Span_arpeggio_engraver` is included in the `Score` context.

```
<<
  \new PianoStaff <<
    \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
    }
  }
>>

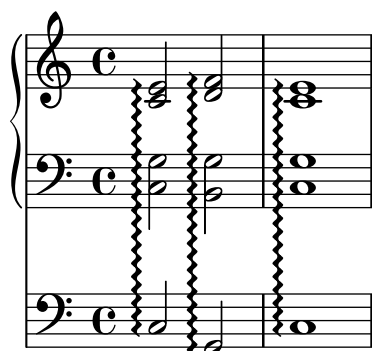
\new Staff \relative c {
  \set Score.connectArpeggios = ##t
```

```

\clef bass
c2\arpeggio g\arpeggio
c1\arpeggio
}
>>

\layout {
  \context {
    \Score
    \consists "Span_arpeggio_engraver"
  }
}

```



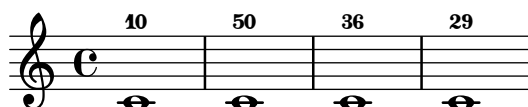
## Creating double-digit fingerings

Creating fingerings larger than 5 is possible.

```

\relative c' {
  c1-10
  c1-50
  c1-36
  c1-29
}

```



## Creating slurs across voices

In some situations it is necessary to create slurs between notes from different voices. The solution is to add invisible notes to one of the voices, using `\hideNotes`.

This example is measure 235 of the Ciaccona from Bach's second partita for solo violin, BWV 1004.

```

\relative c' {
  <<
  {
    d16( a') s a s a[ s a] s a[ s a]
  }
  \\\
  {
    \slurUp
    bes,16[ s e](

```

```

\hideNotes a)
\unHideNotes f[(
\hideNotes a)
\unHideNotes fis](
\hideNotes a)
\unHideNotes g[(
\hideNotes a)
\unHideNotes gis](
\hideNotes a)
}
>>
}

```



## Creating text spanners

The `\startTextSpan` and `\stopTextSpan` commands allow the creation of text spanners as easily as pedal indications or octavations. Override some properties of the `TextSpanner` object to modify its output.

```

\paper { ragged-right = ##f }

\relative c' {
  \override TextSpanner.bound-details.left.text = #"bla"
  \override TextSpanner.bound-details.right.text = #"blu"
  a4 \startTextSpan
  b4 c
  a4 \stopTextSpan

  \override TextSpanner.style = #'line
  \once \override TextSpanner.bound-details.left.stencil-align-dir-y = #CENTER
  a4 \startTextSpan
  b4 c
  a4 \stopTextSpan

  \override TextSpanner.style = #'dashed-line
  \override TextSpanner.bound-details.left.text =
    \markup { \draw-line #'(0 . 1) }
  \override TextSpanner.bound-details.right.text =
    \markup { \draw-line #'(0 . -2) }
  \once \override TextSpanner.bound-details.right.padding = #-2
  a4 \startTextSpan
  b4 c
  a4 \stopTextSpan

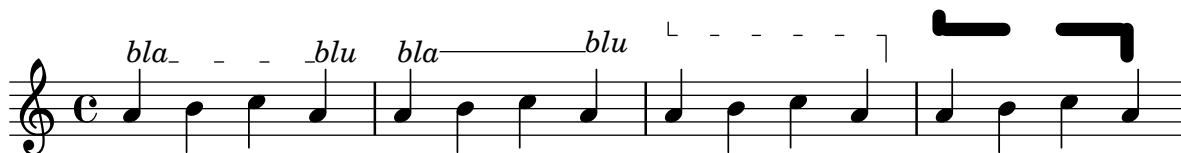
  \override TextSpanner.dash-period = #10
  \override TextSpanner.dash-fraction = #0.5
  \override TextSpanner.thickness = #10
  a4 \startTextSpan

```

```

b4 c
a4 \stopTextSpan
}

```



## Dynamics spanner with custom text

Postfix functions for custom crescendo text spanners. The spanners should start on the first note of the measure. One has to use `-\mycresc`, otherwise the spanner start will rather be assigned to the next note.

```

% Two functions for (de)crescendo spanners where you can explicitly
% give the spanner text.

```

```

mycresc =
#(define-music-function (mymarkup) (markup?)
  (make-music 'CrescendoEvent
    'span-direction START
    'span-type 'text
    'span-text mymarkup))

```

```

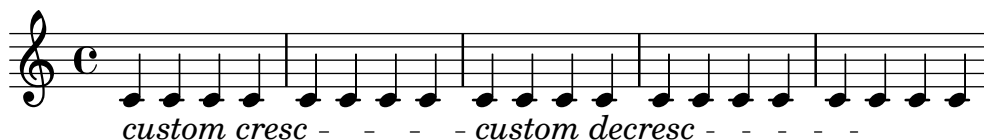
mydecresc =
#(define-music-function (mymarkup) (markup?)
  (make-music 'DecrescendoEvent
    'span-direction START
    'span-type 'text
    'span-text mymarkup))

```

```

\relative c' {
  c4-\mycresc "custom cresc" c4 c4 c4 |
  c4 c4 c4 c4 |
  c4-\mydecresc "custom decresc" c4 c4 c4 |
  c4 c4 c4 c4 |
  c4 c4\! c4 c4
}

```



## Glissandi can skip grobs

NoteColumn grobs can be skipped over by glissandi.

```

\relative c' {
  a2 \glissando
  \once \override NoteColumn.glissando-skip = ##t
  f''4 d,
}

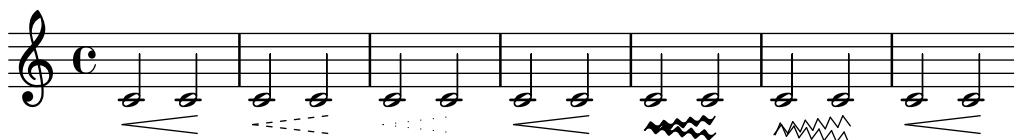
```



## Hairpins with different line styles

Hairpins can take any style from line-interface: dashed-line, dotted-line, line, trill, or zigzag.

```
\relative c' {
  c2\< c\!
  \override Hairpin.style = #'dashed-line
  c2\< c\!
  \override Hairpin.style = #'dotted-line
  c2\< c\!
  \override Hairpin.style = #'line
  c2\< c\!
  \override Hairpin.style = #'trill
  c2\< c\!
  \override Hairpin.style = #'zigzag
  c2\< c\!
  \revert Hairpin.style
  c2\< c\!
}
```



## 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 as follows.

```
\relative c' {
  \override DynamicTextSpanner.style = #'none
  \crescTextCresc
  c1\< | d | b | c\!
}
```



## Horizontally aligning custom dynamics like “più f”

Some dynamic expressions involve additional text, like “sempre **pp**”. Since dynamics are usually centered under the note, the `\pp` would be displayed way after the note it applies to.

To correctly align the “sempre **pp**” horizontally so that it is aligned as if it were only the `\pp`, there are several approaches:

- Simply use `\once\override DynamicText.X-offset = #-9.2` before the note with the dynamics to manually shift it to the correct position. Drawback: This has to be done manually each time you use that dynamic markup...



- Add some padding (`#:hspace 7.1`) into the definition of your custom dynamic mark so that after LilyPond center-aligns it, it is already correctly aligned. Drawback: The padding really takes up that space and does not allow any other markup or dynamics to be shown in that position.
- Shift the dynamic script `\once\override ... .X-offset = ....` Drawback: `\once\override` is needed for every invocation!
- Set the dimensions of the additional text to 0 (using `#:with-dimensions '(0 . 0) '(0 . 0)`). Drawback: For LilyPond, “sempre” has no extent now. This means it might put other stuff there, causing collisions (which are not detected by LilyPond’s collision detection algorithm!). There also seems to be some spacing, so it is not exactly the same alignment as without the additional text.
- Add an explicit shift directly inside the scheme function for the dynamic script.
- Set an explicit alignment inside the dynamic script. By default, this won’t have any effect, only if one sets `X-offset`! Drawback: One needs to set `DynamicText.X-offset`, which will apply to all dynamic texts! Also, it is aligned at the right edge of the additional text, not at the center of `\pp`.

```
\paper {
  ragged-right = ##f
  indent = 5\cm
}
```

```
% Solution 1: Using a simple markup with a particular halign value
% Drawback: It's a markup, not a dynamic command, so \dynamicDown
%           etc. will have no effect
semppMarkup = \markup { \halign #1.4 \italic "sempre" \dynamic "pp" }
```

```
% Solution 2: Using a dynamic script & shifting with
%           \once \override ...X-offset = ..
% Drawback: \once \override needed for every invocation
semppK =
#(make-dynamic-script
  (markup #:line
    (#:normal-text
      #:italic "sempre"
      #:dynamic "pp"))))
```

```
% Solution 3: Padding the dynamic script so the center-alignment
%           puts it at the correct position
% Drawback: the padding really reserves the space, nothing else can be there
semppT =
#(make-dynamic-script
  (markup #:line
    (#:normal-text
      #:italic "sempre"
      #:dynamic "pp"
      #:hspace 7.1))))
```

```
% Solution 4: Dynamic, setting the dimensions of the additional text to 0
% Drawback: To lilypond "sempre" has no extent, so it might put
%           other stuff there => collisions
% Drawback: Also, there seems to be some spacing, so it's not exactly the
```

```

%           same alignment as without the additional text
sempM =
#(make-dynamic-script
  (markup #:line
    (:with-dimensions '(0 . 0) '(0 . 0)
      #:right-align
      #:normal-text
      #:italic "sempre"
      #:dynamic "pp")))

% Solution 5: Dynamic with explicit shifting inside the scheme function
sempG =
#(make-dynamic-script
  (markup #:hspace 0
    #:translate '(-18.85 . 0)
    #:line (:normal-text
      #:italic "sempre"
      #:dynamic "pp")))

% Solution 6: Dynamic with explicit alignment. This has only effect
%           if one sets X-offset!
% Drawback: One needs to set DynamicText.X-offset!
% Drawback: Aligned at the right edge of the additional text,
%           not at the center of pp
sempMII =
#(make-dynamic-script
  (markup #:line (:right-align
    #:normal-text
    #:italic "sempre"
    #:dynamic "pp")))

\new StaffGroup <<
  \new Staff \with { instrumentName = "standard" }
    \relative c'' {
      \key es \major
      c4\pp c\p c c | c\ff c c\pp c
    }
  \new Staff \with {instrumentName = "normal markup" }
    \relative c'' {
      \key es \major
      c4-\sempMarkup c\p c c | c\ff c c-\sempMarkup c
    }
  \new Staff \with { instrumentName = "explicit shifting" }
    \relative c'' {
      \key es \major
      \once \override DynamicText.X-offset = #-9.2
      c4\sempK c\p c c
      c4\ff c
      \once \override DynamicText.X-offset = #-9.2
      c4\sempK c
    }
  \new Staff \with { instrumentName = "right padding" }

```

```

\relative c'' {
  \key es \major
  c4\semppT c\p c c | c\ff c c\semppT c
}
\new Staff \with { instrumentName = "set dimension to zero" }
\relative c'' {
  \key es \major
  c4\semppM c\p c c | c\ff c c\semppM c
}
\new Staff \with { instrumentName = "shift inside dynamics" }
\relative c'' {
  \key es \major
  c4\semppG c\p c c | c\ff c c\semppG c
}
\new Staff \with { instrumentName = "alignment inside dynamics" }
\relative c'' {
  \key es \major
  \override DynamicText.X-offset = #-1
  c4\semppMII c\p c c | c\ff c c\semppMII c
}
>>

\layout { \override Staff.InstrumentName.self-alignment-X = #LEFT }

```

standard

normal markup

explicit shifting

right padding

set dimension to zero

shift inside dynamics

alignment inside dynamics

## 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
}
```



## Laissez vibrer ties

Laissez vibrer ties have a fixed size. Their positioning can be tuned using the tie-configuration property.

See also snippet “Longer laissez vibrer ties”.

```
\relative c' {
  <c e g>4\laissezVibrer r <c f g>\laissezVibrer r
  <c d f g>4\laissezVibrer r <c d f g>4.\laissezVibrer r8

  <c d e f>4\laissezVibrer r
  \override LaissezVibrerTieColumn.tie-configuration
    = #`((-7 . ,DOWN)
        (-5 . ,DOWN)
        (-3 . ,UP)
        (-1 . ,UP))
  <c d e f>4\laissezVibrer r
}
```



## Line arrows

Arrows can be applied to text spanners and line spanners (such as glissandi).

```
\relative c' ' {
  \override TextSpanner.bound-padding = #1.0
  \override TextSpanner.style = #'line
  \override TextSpanner.bound-details.right.arrow = ##t
  \override TextSpanner.bound-details.left.text = #"fof"
  \override TextSpanner.bound-details.right.text = #"gag"
  \override TextSpanner.bound-details.right.padding = #0.6

  \override TextSpanner.bound-details.right.stencil-align-dir-y = #CENTER
  \override TextSpanner.bound-details.left.stencil-align-dir-y = #CENTER
```

```

\override Glissando.bound-details.right.arrow = ##t
\override Glissando.arrow-length = #0.5
\override Glissando.arrow-width = #0.25

a8\startTextSpan gis a4 b\glissando b,
g'4 c\stopTextSpan c2
}

```



## Making slurs with complex dash structure

Slurs can be composed of complex dash patterns by setting the `dash-definition` property, which is a list of slur segments, which in turn are lists of parameters setting up the dash behavior of the given segment.

Slur segments are defined in terms of the Bézier parameter  $t$ , which ranges from 0 at the left end of the slur to 1 at the right end of the slur. A slur segment has the form  $(start-t\ stop-t\ dash-fraction\ dash-period)$ . In the segment spanning the range  $start-t$  to  $stop-t$ , the dash pattern is defined by the values of  $dash-fraction$  and  $dash-period$ .  $dash-fraction$  specifies how much of a dash period is black; if set to 1 you get a solid slur segment. The unit for  $dash-period$  is staff spaces.

```

\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)
}

```



## 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`:

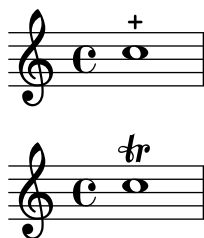
```

\relative c' { c1-+ }

dashPlus = \trill

```

```
\relative c' ' { c1-+ }
```

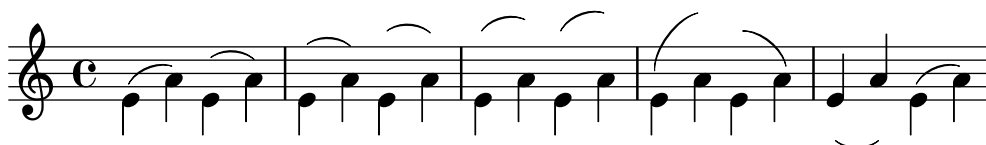


## Moving slur positions vertically

The vertical position of a slur can be adjusted using the `positions` property of `Slur`. The property has 2 parameters, the first referring to the left end of the slur and the second to the right. The values of the parameters are not used by LilyPond to make an exact movement of the slur - instead it selects what placement of the slur looks best, taking into account the parameter values. Positive values move the slur up, and are appropriate for notes with stems down. Negative values move downward slurs further down.

See also “Adjusting slur positions vertically”.

```
\relative c' {
  \stemDown
  e4( a)
  \override Slur.positions = #'(1 . 1)
  e4( a)
  \override Slur.positions = #'(2 . 2)
  e4( a)
  \override Slur.positions = #'(3 . 3)
  e4( a)
  \override Slur.positions = #'(4 . 4)
  e4( a)
  \override Slur.positions = #'(5 . 5)
  e4( a)
  \override Slur.positions = #'(0 . 5)
  e4( a)
  \override Slur.positions = #'(5 . 0)
  e4( a)
  \stemUp
  \override Slur.positions = #'(-5 . -5)
  e4( a)
  \stemDown
  \revert Slur.positions
  e4( a)
}
```



## Moving the ends of hairpins

The ends of hairpins may be offset by setting the `shorten-pair` property of the `Hairpin` object. Positive values move endpoints to the right, negative to the left. Unlike the `minimum-length`

property, this property only affects the appearance of the hairpin; it does not adjust horizontal spacing (including the position of bounding dynamics). This method is thus suitable for fine-tuning a hairpin within its allotted space.

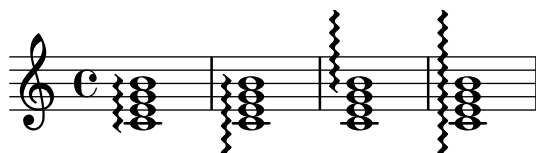
```
{
  c'1~\<
  c'2~ c'\!
  \once \override Hairpin.shorten-pair = #'(2 . 2)
  c'1~\<
  c'2~ c'\!
  \once \override Hairpin.shorten-pair = #'(-2 . -2)
  c'1~\<
  c'2~ c'\!
  c'1~\p-\tweak shorten-pair #'(2 . 0)\<
  c'2~ c'\ffff
}
```



## Positioning arpeggios

If you need to extend or shorten an arpeggio, you can modify the upper and lower start positions independently.

```
\relative c' {
  <c e g b>1\arpeggio
  \once \override Arpeggio.positions = #'(-5 . 0)
  <c e g b>1\arpeggio
  \once \override Arpeggio.positions = #'(0 . 5)
  <c e g b>1\arpeggio
  \once \override Arpeggio.positions = #'(-5 . 5)
  <c e g b>1\arpeggio
}
```



## Positioning text markups inside slurs

Text markups need to have the outside-staff-priority property set to #f 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
}
```



## Printing hairpins in various styles

Hairpin dynamics may be created in a variety of styles.

```
\relative c' ' {
  \override Hairpin.stencil = #flared-hairpin
  a4\< a a a\f
  a4\p\< a a a\ff
  a4\sfz\< a a a\!
  \override Hairpin.stencil = #constante-hairpin
  a4\< a a a\f
  a4\p\< a a a\ff
  a4\sfz\< a a a\!
  \override Hairpin.stencil = #flared-hairpin
  a4\> a a a\f
  a4\p\> a a a\ff
  a4\sfz\> a a a\!
  \override Hairpin.stencil = #constante-hairpin
  a4\> a a a\f
  a4\p\> a a a\ff
  a4\sfz\> a a a\!
}
```



## Printing hairpins using al niente notation

Hairpin dynamics may be printed with a circled tip (“al niente” notation) by setting the circled-tip property of the Hairpin object to #t.

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





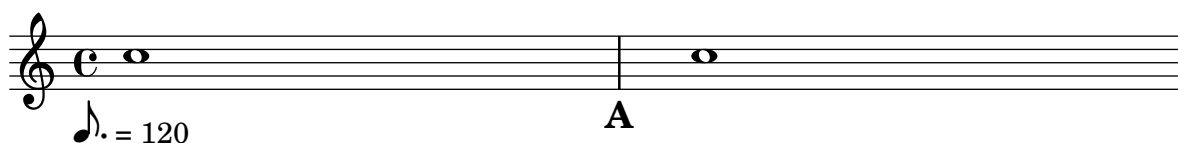
## Printing metronome and rehearsal marks below the staff

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

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

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

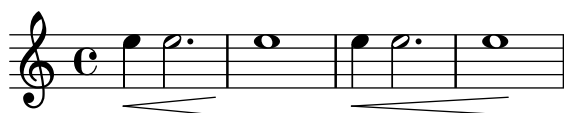
  % Rehearsal marks below the staff
  \override Score.RehearsalMark.direction = #DOWN
  \mark \default
  c''1
}
```



## Setting hairpin behavior at bar lines

If the note which ends a hairpin falls on a downbeat, the hairpin stops at the bar line immediately preceding. This behavior can be controlled by overriding the `to-barline` property.

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

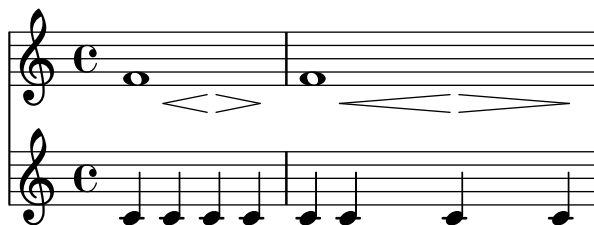


## Setting the minimum length of hairpins

If hairpins are too short, they can be lengthened by modifying the `minimum-length` property of the `Hairpin` object.

```
<<
{
  \after 4 \< \after 2 \> \after 2. \! f'1
  \override Hairpin.minimum-length = 8
  \after 4 \< \after 2 \> \after 2. \! f'1
}
{
  \repeat unfold 8 c'4
}
```

```
}
>>
```



## Showing the same articulation above and below a note or chord

By default, LilyPond does not allow the same articulation (an accent, a fermata, a flageolet, etc.) to be displayed above and below a note. For example, `c4_\fermata^\fermata` only shows a fermata below. The fermata above gets simply ignored.

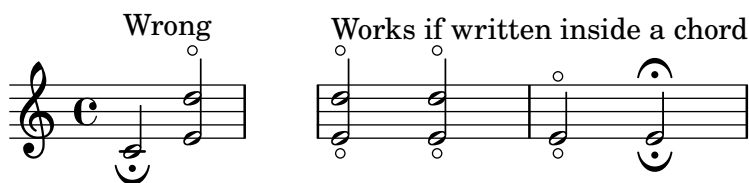
However, one can stick scripts (just like fingerings) inside a chord, which means it is possible to have as many articulations as desired. This approach has the advantage that it ignores the stem and positions the articulation relative to the note head. This can be seen in the case of the flageolets in the snippet. To mimic the behaviour of scripts outside a chord, `add-stem-support` would be required.

The solution is thus to write the note as a chord and add the articulations inside of `<...>`, using the direction modifiers `^` and `_` as appropriate.

```
\relative c' {
  <>^"Wrong"
  c2_\fermata^\fermata % The second fermata is ignored!
  <e d'>2^\flageolet_\flageolet

  \stopStaff s1 \startStaff

  <>^"Works if written inside a chord"
  <e_\flageolet d'^\flageolet>2
  <e_\flageolet d'^\flageolet>2
  <e_\flageolet^\flageolet>2
  <e_\fermata^\fermata>2
}
```

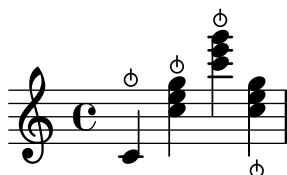


## Snap pizzicato (“Bartok” pizzicato)

A snap pizzicato (also known as “Bartok pizzicato”) is a “strong pizzicato where the string is plucked vertically by snapping and rebounds off the fingerboard of the instrument” (Wikipedia). It is denoted by a circle with a vertical line going from the center upwards outside the circle.

```
\relative c' {
  c4\snappizzicato
  <c' e g>4\snappizzicato
  <c' e g>4^\snappizzicato
  <c, e g>4_\snappizzicato
```

}

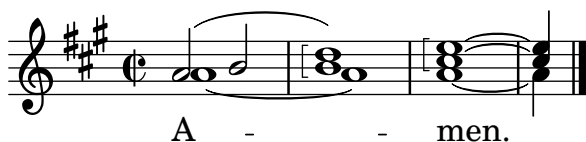


## Using `\arpeggioBracket` to make divisi more visible

The `\arpeggioBracket` command can be used to indicate the division of voices where there are no stems to provide the information. This is often seen in choral music.

```
\include "english.ly"
```

```
\score {
  \relative c'' {
    \key a \major
    \time 2/2
    <<
      \new Voice = "upper"
      <<
        { \voiceOne \arpeggioBracket
          a2( b2
            <b d>1\arpeggio)
            <cs e>\arpeggio ~
            <cs e>4
          }
        \addlyrics { \lyricmode { A -- men. } }
      >>
      \new Voice = "lower"
      { \voiceTwo
        a1 ~
        a
        a ~
        a4 \bar "|"
      }
    >>
  }
}
```



## 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 staff.

```
\relative c'' {
  c2
```

```

\breathes
d2
\override BreathingSign.Y-offset = #2.6
\override BreathingSign.text =
  \markup { \musicglyph "scripts.tickmark" }
c2
\breathes
d2
}

```



## Using double slurs for legato chords

Some composers write two *slurs* when they want legato chords. This can be achieved by setting `doubleSlurs` context property.

```

\relative c' {
  \set doubleSlurs = ##t
  <c e>4( <d f> <c e> <d f>)
}

```



## Using the whiteout property

Any graphical object can be printed over a white background to mask parts of objects that lie beneath. This can be useful to improve the appearance of collisions in complex situations when repositioning objects is impractical. It is necessary to explicitly set the `layer` property to control which objects are masked by the white background.

In this example the collision of the tie with the time signature is improved by masking out the part of the tie that crosses the time signature, setting the `whiteout` property of `TimeSignature`. To do this, `TimeSignature` is moved to a layer above `Tie`, which is left in the default layer 1, and `StaffSymbol` is moved to a layer above `TimeSignature` so it is not masked.

```

{
  \override Score.StaffSymbol.layer = 4
  \override Staff.TimeSignature.layer = 3
  b'2 b'~
  \once \override Staff.TimeSignature.whiteout = ##t
  \time 3/4
  b' r4
}

```

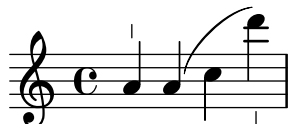


## Vertical line as a baroque articulation mark

This short vertical line placed above the note is commonly used in baroque music. Its meaning can vary, but generally indicates notes that should be played with more “weight”. The following example demonstrates how to achieve such a notation.

```
upline =
\tweak stencil
  #(\lambda (grob)
    (grob-interpret-markup grob #{ \markup \draw-line #'(0 . 1) #}))
  \stopped

\relative c' {
  a'4^\upline a( c d')_\upline
}
```



## Vertically aligning dynamics across multiple notes

Dynamics that occur at, begin on, or end on the same note will be vertically aligned. To ensure that dynamics are aligned when they do not occur on the same note, increase the `staff-padding` property of the `DynamicLineSpanner` object.

```
\relative c' {
  \override DynamicLineSpanner.staff-padding = #4
  c2\p f\mf
  g2\< b4\> c\!
}
```



## 4 Repeats

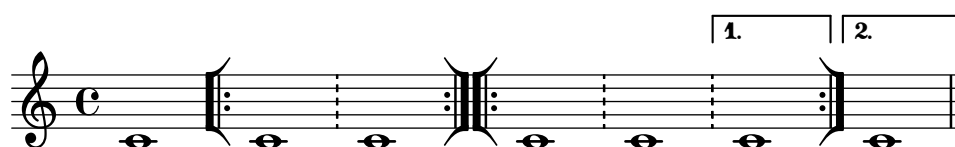
See also Section “Repeats” in *Notation Reference*.

### Changing the default bar lines

Default bar lines can be changed when re-defined in a Score context.

```
\layout {
  \context {
    \Score
    % Changing the defaults from engraver-init.ly
    measureBarType = "!"
    startRepeatBarType = "[|:"
    endRepeatBarType = ":|]"
    doubleRepeatBarType = ":||[|:"
  }
}

{
  c'1
  \repeat volta 2 { c' c' }
  \repeat volta 2 { c' c' \alternative { \volta 1 { c' }
                                         \volta 2 { c' } } }
  \bar "|."
}
```



### Controlling the appearance of tremolo slashes

Using various properties of the StemTremolo grob it is possible to control the appearance of tremolo slashes.

- Property `slope` sets the slope for tremolo slashes.
- Property `shape` determines whether tremolo slashes look like rectangles (value `rectangle`) or like very small beams (value `beam-like`).
- Property `style` sets both the slope and the shape depending on whether the note has flags, beams, or only a plain stem. This is in contrast to the previous two properties, which change the slope and shape unconditionally. There are two styles defined.
  - `default`: slashes for down-stem flags are longer and more sloped than slashes for up-stem flags; slashes on beamed notes have a rectangular shape and are parallel to the beam.
  - `constant`: all slashes are beam-like and have the same slope except for down-stem flags.

```
music = {
  a''4:32 a':
  e''8: \noBeam e':
  a'':[ a':]
  f':[ g':]
```

```

d':[ d':]
}

\new Staff {
  <>^\markup "default"
  \music
}

\new Staff {
  <>^\markup \typewriter "style = #'constant"
  \override StemTremolo.style = #'constant
  \music
}

\new Staff {
  <>^\markup \typewriter "shape = #'rectangle"
  \override StemTremolo.shape = #'rectangle
  \music
}

\new Staff {
  <>^\markup \typewriter "shape = #'beam-like"
  \override StemTremolo.shape = #'beam-like
  \music
}

\new Staff {
  <>^\markup \typewriter "slope = -0.2"
  \override StemTremolo.slope = -0.2
  \music
}

```





## 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
      }
    }
  }
>>
```



## Engraving tremolos with floating beams

If a tremolo's total duration is less than a quarter-note, or exactly a half note, or between a half note and a whole note, it is normally typeset with all beams touching the stems. Certain engraving styles typeset some of these beams as centered floating beams that do not touch the stems. The number of floating beams in this type of tremolo is controlled with the `gap-count` property of the `Beam` object, and the size of the gaps between beams and stems is set with the `gap` property.

```
\relative c'' {
  \repeat tremolo 8 { a32 f }
  \override Beam.gap-count = #1
  \repeat tremolo 8 { a32 f }
  \override Beam.gap-count = #2
}
```



```

\repeat tremolo 8 { a32 f }
\override Beam.gap-count = #3
\repeat tremolo 8 { a32 f }

\override Beam.gap-count = #3
\override Beam.gap = #1.33
\repeat tremolo 8 { a32 f }
\override Beam.gap = #1
\repeat tremolo 8 { a32 f }
\override Beam.gap = #0.67
\repeat tremolo 8 { a32 f }
\override Beam.gap = #0.33
\repeat tremolo 8 { a32 f }
}

```



## Isolated percent repeats

Isolated percents can also be printed.

```

makePercent =
#(define-music-function (note) (ly:music?)
  "Make a percent repeat the same length as NOTE."
  (make-music 'PercentEvent
    'length (ly:music-length note)))

\relative c' {
  \makePercent s1
}

```



## Measure counters

This snippet demonstrates the use of the `Measure_counter_engraver` to number groups of successive measures. Any stretch of measures may be numbered, whether consisting of repetitions or not.

The engraver must be added to the appropriate context. Here, a `Staff` context is used; another possibility is a `Dynamics` context.

The counter is begun with `\startMeasureCount` and ended with `\stopMeasureCount`. Numbering will start by default with 1, but this behavior may be modified by overriding the `count-from` property.

When a measure extends across a line break, the number will appear twice, the second time in parentheses.

```

\layout {
  \context {
    \Staff
    \consists #Measure_counter_engraver

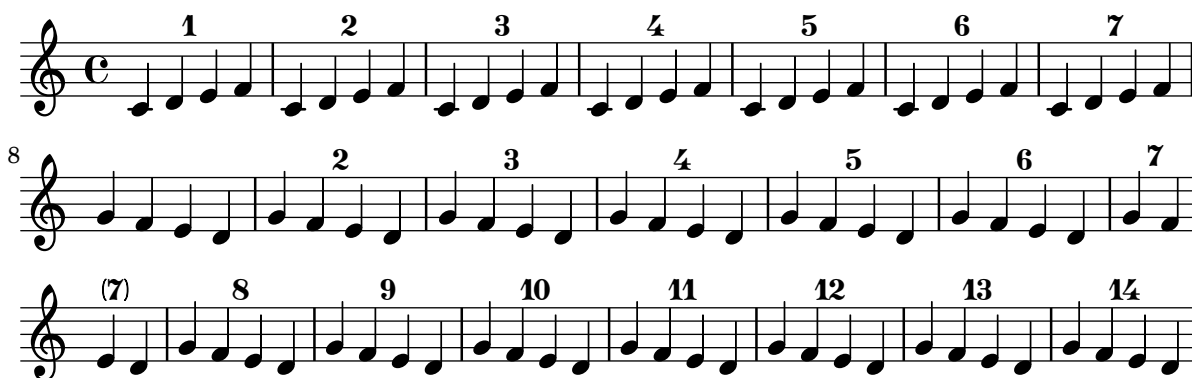
```

```

}
}

\new Staff {
  \startMeasureCount
  \repeat unfold 7 {
    c'4 d' e' f'
  }
  \stopMeasureCount
  \bar "||"
  g'4 f' e' d'
  \override Staff.MeasureCounter.count-from = #2
  \startMeasureCount
  \repeat unfold 5 {
    g'4 f' e' d'
  }
  g'4 f'
  \bar ""
  \break
  e'4 d'
  \repeat unfold 7 {
    g'4 f' e' d'
  }
  \stopMeasureCount
}

```



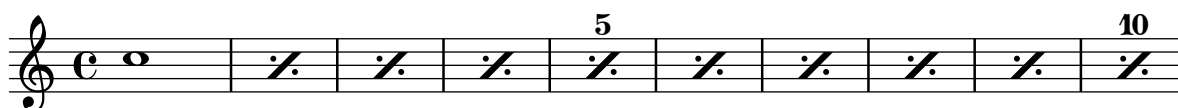
## Percent repeat count visibility

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 }
}

```





## Percent repeat counter

Measure repeats of more than two repeats are printed with a counter if the `countPercentRepeats` context property is set.

```
\relative c' {
  \set countPercentRepeats = ##t
  \repeat percent 4 { c1 }
}
```



## Positioning segno and coda (with line break)

If you want to place an exiting segno sign and add text like “D.S. al Coda” next to it where usually the staff lines are you can use this snippet. The coda will resume in a new line. There is a variation documented in this snippet, where the coda will remain on the same line.

```
\relative c' {
  c4 c c c | c c c c |
  \repeat segno 2 {
    c4 c c c | c c c c |
    \alternative {
      \volta 1 {
        c4 c c c | c c c c |
        % If you don't use \break at Coda, use \noBreak here
        % and after \bar "" below.
        \noBreak
        \section % double bar line
        \cadenzaOn % pause bar count
        \stopStaff % remove staff lines
        % Increasing the unfold counter will expand the staff-free space
        \repeat unfold 4 {
          s1
          \bar ""
        }
        % Place JumpScript where the staff would normally be.
        \once \override Score.JumpScript.outside-staff-priority = ##f
        \once \override Score.JumpScript.Y-offset = 0
        \startStaff % resume bar count
        \cadenzaOff % show staff lines again
      }
    }
  }
}
\sectionLabel "Coda"
% Show Coda on a new line
\break
\repeat unfold 6 { c4 c c c }
```

```
\fine
}
```

(7) Coda

## Setting the double repeat default for volte

There are different double repeat styles for volte that can be selected using the context property `doubleRepeatBarType`.

```
\relative c' {
  \repeat volta 2 { c1 }
  \set Score.doubleRepeatBarType = ":\dots:"
  \repeat volta 2 { c1 }
  \set Score.doubleRepeatBarType = ":\|.|\:|"
  \repeat volta 2 { c1 }
  \set Score.doubleRepeatBarType = ":\|.|\:|"
  \repeat volta 2 { c1 }
}
```

## 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 overriding `VoltaBracket.musical-length`. In the next example, the bracket only lasts one measure, which is a duration of 3/4.

```
\fixed c' {
  \time 3/4
  c4 c c
  \repeat volta 5 {
    d4 d d
    \alternative {
      \volta 1,2,3,4 {
        \once \override Score.VoltaBracket.musical-length =
        \musicLength 2.
        e4 e e
        f4 f f
      }
      \volta 5 {
        g4 g g
        g2.
      }
    }
  }
}
```

}



## Unfolding tremolo repeats

Currently, `note:duration`, which is a more or less a shortcut for `\repeat tremolo`, is not unfolded by `\unfoldRepeats` (this is tracked in issue #6145 (<https://gitlab.com/lilypond/lilypond/-/issues/6145>)). The function given in this snippet provides a workaround.

```
fixTremolos =
#(define-music-function (music) (ly:music?)
  (music-map
    (lambda (m)
      (let ((event (any (lambda (a)
                          (and (music-is-of-type? a 'tremolo-event)
                              a))
                        (ly:music-property m 'articulations))))
        (if event
          (let* ((total-tremolo-duration (ly:music-property m
                                                             'duration))
                 (tremolo-type (ly:music-property event
                                                    'tremolo-type))
                 (one-tremolo-note-duration
                  (ly:make-duration (ly:intlog2 tremolo-type)))
                 (tremolo-note-count
                  (/ tremolo-type (expt 2 (ly:duration-log
                                           total-tremolo-duration)))))
            (set! (ly:music-property m 'duration)
                  one-tremolo-note-duration)
            (set! (ly:music-property m 'articulations)
                  (delete! event (ly:music-property m 'articulations)))
            (make-music 'TremoloRepeatedMusic
                        'repeat-count tremolo-note-count
                        'element m))
          m)))
    music))

unfoldRepeats = \unfoldRepeats #'() \fixTremolos \etc

music = { \repeat tremolo 8 c'16 c'2:16 }

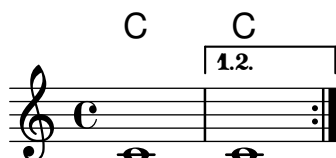
{
  \music
  \unfoldRepeats \music
}
```



## Volta below chords

By adding the `Volta_engraver` to the relevant staff, volte can be put below chords.

```
\score {
  <<
    \chords { c1 c1 }
    \new Staff \with { \consists "Volta_engraver" }
    {
      \repeat volta 2 { c'1 \alternative { c' } }
    }
  >>
  \layout {
    \context {
      \Score
      \remove "Volta_engraver"
    }
  }
}
```



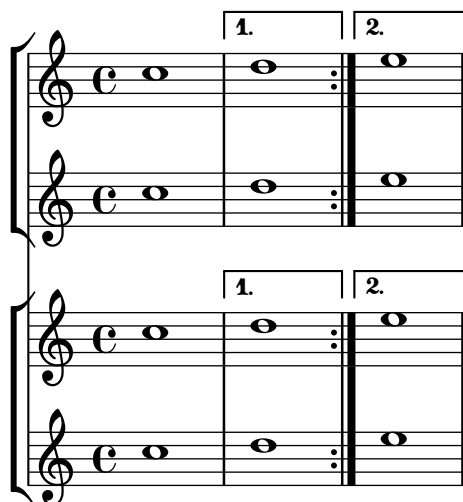
## Volta brackets in multiple staves

By adding the `Volta_engraver` to the relevant staff, volte can be put over staves other than the topmost one in a score.

`\repeat` and related commands should be present in all staves.

```
voltaMusic = \relative c'' {
  \repeat volta 2 {
    c1
    \alternative {
      \volta 1 { d1 }
      \volta 2 { e1 }
    }
  }
}

<<
\new StaffGroup <<
  \new Staff \voltaMusic
  \new Staff \voltaMusic
>>
\new StaffGroup <<
  \new Staff \with { \consists "Volta_engraver" }
    \voltaMusic
  \new Staff \voltaMusic
>>
>>
```



## Volta text markup using repeatCommands

Though voltes are best specified using `\repeat volta`, the context property `repeatCommands` must be used in cases where the volta text needs more advanced formatting with `\markup`.

Since `repeatCommands` takes a list, the simplest method of including markup is to use an identifier for the text and embed it in the command list using the Scheme syntax `#(list (list 'volta textIdentifier))`. Start- and end-repeat commands can be added as separate list elements:

```
voltaAdLib = \markup { \volta-number { 1. 2. 3... } \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))
}
```



## 5 Simultaneous notes

See also Section “Simultaneous notes” in *Notation Reference*.

### Additional voices to avoid collisions

In some instances of complex polyphonic music, additional voices are necessary to prevent collisions between notes. If more than four parallel voices are needed, additional voices can be added by defining a variable using the Scheme function `context-spec-music`.

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

```
\relative c' ' {
  \time 3/4
  \key d \minor
  \partial 2
  <<
    \new Voice {
      \voiceOne
      a4. a8
      e'4 e4. e8
      f4 d4. c8
    }
    \new Voice {
      \voiceTwo
      d,2
      d4 cis2
      d4 bes2
    }
    \new Voice {
      \voiceThree
      f'2
      bes4 a2
      a4 s2
    }
    \new Voice {
      \voiceFive
      s2
      g4 g2
      f4 f2
    }
  >>
}
```



### 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.soloIIIText = "boy"
\set Staff.aDueText = "together"
\partCombine
  \relative c'' {
    g4 g r r
    a2 g
  }
  \relative c'' {
    r4 r a( b)
    a2 g
  }
>>

```



## Changing a single note's size in a chord

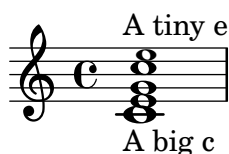
Individual note heads in a chord can be modified with the `\tweak` command inside a chord, by altering the `font-size` property.

Inside the chord (within the brackets `< >`), before the note to be altered, place the `\tweak` command, followed by `font-size` and define the proper size like `#-2` (a tiny note head).

```

\relative c' {
  <\tweak font-size #-2 c e g c
  \tweak font-size #-2 e>1
  ~\markup { A tiny e }_ \markup { A big c }
}

```



## Clusters

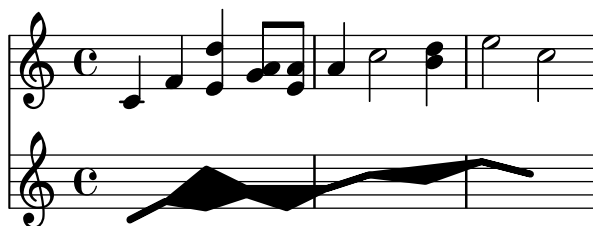
Clusters are a device to denote that a complete range of notes is to be played.

```

fragment = \relative c' {
  c4 f <e d'>4
  <g a>8 <e a> a4 c2 <d b>4
  e2 c
}

<<
  \new Staff \fragment
  \new Staff \makeClusters \fragment
>>

```



## Combining two parts on the same staff

The part combiner tool (i.e., the `\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 them 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.

```
musicUp = \relative c'' {
  \time 4/4
  a4 c4.( g8) a4 |
  g4 e' g,( a8 b) |
  c b a2.
}

musicDown = \relative c'' {
  g4 e4.( d8) c4 |
  r2 g'4( f8 e) |
  d2 \stemDown a
}

\score {
  <<
    \new Staff \with {
      instrumentName = "standard polyphony"
    } << \musicUp \\\musicDown >>

    \new Staff \with {
      instrumentName =
        \markup { \typewriter "\\partCombine" without text}
      printPartCombineTexts = ##f
    } \partCombine \musicUp \musicDown

    \new Staff \with {
      instrumentName =
        \markup { \typewriter "\\partCombine" with text}
    } \partCombine \musicUp \musicDown
  >>

  \layout {
    indent = 6.0\cm
    \context {
```

```

\Score
% Setting this to a large value avoids a bar line at the
% beginning that would connect the three staves otherwise.
\override SystemStartBar.collapse-height = 30
}
}
}

```

standard polyphony

\partCombine without text

\partCombine with text



## Displaying complex chords

Here is a way to display a chord where the same note is played twice with different accidentals.

```

fixA = {
  \once \override Stem.length = #12
}

fixB = {
  \once \override NoteHead.X-offset = #1.7
  \once \override Stem.length = #7
  \once \override Stem.rotation = #'(45 0 0)
  \once \override Stem.extra-offset = #'(-0.1 . -0.2)
  \once \override Flag.style = #'no-flag
  \once \override Accidental.extra-offset = #'(4 . -.1)
}

\relative c' {
  << { \fixA <b d!>8 } \ { \voiceThree \fixB dis } >> s
}

```



## Forcing horizontal shift of notes

When the typesetting engine cannot cope, the following syntax can be used to override typesetting decisions. The units of measure used here are staff spaces.

```

\relative c' <<
{
  <d g>2 <d g>
}

```

```

}
\\
{
  <b f'>2
  \once \override NoteColumn.force-hshift = 1.7
  <b f'>2
}
>>

```



## Making an object invisible using the ‘transparent’ property

Setting the transparent property will cause an object to be printed in “invisible ink”: the object is not printed, but all its other behavior is retained. The object still takes up space, it takes part in collisions, and slurs, ties and beams can be attached to it.

This snippet demonstrates how to connect different voices using ties. Normally, ties only connect two notes in the same voice. By introducing a tie in a different voice, and blanking the first up-stem in that voice, the tie appears to cross voices.

```

\relative {
  \time 2/4
  <<
  {
    \once \hide Stem
    \once \override Stem.length = #8
    b'8 ~ 8\noBeam
    \once \hide Stem
    \once \override Stem.length = #8
    g8 ~ 8\noBeam
  }
  \\
  {
    b8 g g e
  }
  >>
}

\paper {
  line-width = 40\mm
  ragged-right = ##f
}

```



## Moving dotted notes in polyphony

When a dotted note in the upper voice is moved to avoid a collision with a note in another voice, the default is to move the upper note to the right. This behaviour can be over-ridden by using the `prefer-dotted-right` property of `NoteCollision`.

```
\new Staff \relative c' <<
{
  f2. f4
  \override Staff.NoteCollision.prefer-dotted-right = ##f
  f2. f4
  \override Staff.NoteCollision.prefer-dotted-right = ##t
  f2. f4
}
\\
{ e4 e e e e e e e e e e }
>>
```



## Suppressing warnings for clashing note columns

If notes from two voices with stems in the same direction are placed at the same position, and both voices have no shift or the same shift specified, the error message “warning: ignoring too many clashing note columns” appears when compiling the LilyPond file. This message can be suppressed by setting the `ignore-collision` property of the `NoteColumn` object to `#t`. Please note that this does not just suppress warnings but stops LilyPond trying to resolve collisions at all and so may have unintended results unless used with care.

```
ignore = \override NoteColumn.ignore-collision = ##t
```

```
\relative c' {
  \new Staff <<
    \new Voice { \ignore \stemDown f2 g }
    \new Voice { c2 \stemDown c, }
  >>
}
```



## Two \partCombine pairs on one staff

The `\partCombine` function takes two music expressions, each containing a part, and distributes them among four `Voice` contexts named “one”, “two”, “solo”, and “shared”, depending on when and how the parts are merged into a common voice.

Variants of `\partCombine` are `\partCombineUp` and `\partCombineDown` to produce up-stem and down-stem merging of two voices, respectively. Combining them to squeeze four parts into a single staff, however, need some special setup, which this snippet defines accordingly.

```
customPartCombineUp =
#(define-music-function (part1 part2) (ly:music? ly:music?)
```

"Make an up-stem `VoiceBox` context that combines PART1 and PART2.

The context is called 'Up'; internally, the function calls  
`\\partCombineUp`.`"

```
{
  \new VoiceBox = "Up" <<
    \context Voice = "one" { \voiceOne }
    \context Voice = "two" { \voiceThree }
    \context Voice = "shared" { \voiceOne }
    \context Voice = "solo" { \voiceOne }
    \context NullVoice = "null" {}
    \partCombine #part1 #part2
  >>
}
```

`customPartCombineDown =`

```
#(define-music-function (part3 part4) (ly:music? ly:music?)
  "Make a down-stem `VoiceBox` context that combines PART3 and PART4.
```

The context is called 'Down'; internally, the function calls  
`\\partCombineDown`.`"

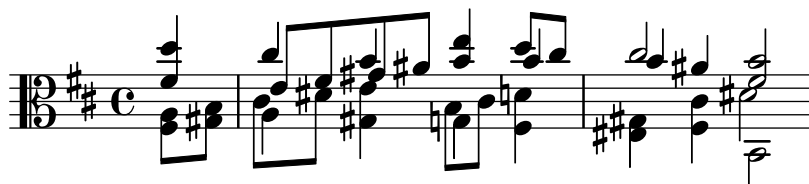
```
{
  \new VoiceBox = "Down" <<
    \set VoiceBox.soloText = #"Solo III"
    \set VoiceBox.soloIIText = #"Solo IV"
    \context Voice = "one" { \voiceFour }
    \context Voice = "two" { \voiceTwo }
    \context Voice = "shared" { \voiceFour }
    \context Voice = "solo" { \voiceFour }
    \context NullVoice = "null" {}
    \partCombine #part3 #part4
  >>
}
```

```
soprano = { d'4 | cis' b e' d'8 cis' | cis'2 b }
alto = { fis4 | e8 fis gis ais b4 b | b ais fis2 }
tenor = { a8 b | cis' dis' e'4 b8 cis' d'4 | gis cis' dis'2 }
bass = { fis8 gis | a4 gis g fis | eis fis b,2 }
```

```
\new Staff <<
  \key b\minor
  \clef alto
  \partial 4
  \transpose b b' \customPartCombineUp \soprano \alto
  \customPartCombineDown \tenor \bass
>>
```

```
\layout {
  \context {
    \Staff
    \accepts "VoiceBox"
  }
}
```

```
\context {  
  \name "VoiceBox"  
  \type "Engraver_group"  
  \defaultchild "Voice"  
  \accepts "Voice"  
  \accepts "NullVoice"  
}  
}
```



## 6 Staff notation

See also Section “Staff notation” in *Notation Reference*.

### Adding ambitus per voice

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

```
\new Staff <<
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c'' {
    \override Ambitus.X-offset = 2.0
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c' {
    \voiceTwo
    es4 f g as
    b1
  }
}>>
```



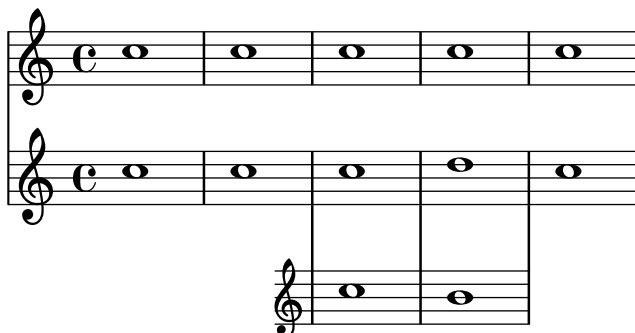
### Adding an extra staff

An extra staff can be added (possibly temporarily) after the start of a piece.

```
\score {
  <<
    \new Staff \relative c'' {
      c1 | c | c | c | c
    }
    \new StaffGroup \relative c'' {
      \new Staff {
        c1 | c
      } <<
        { c1 | d }
      \new Staff {
        \once \omit Staff.TimeSignature
        c1 | b
      }
    } >>
  } >>
  c1
}
```



}

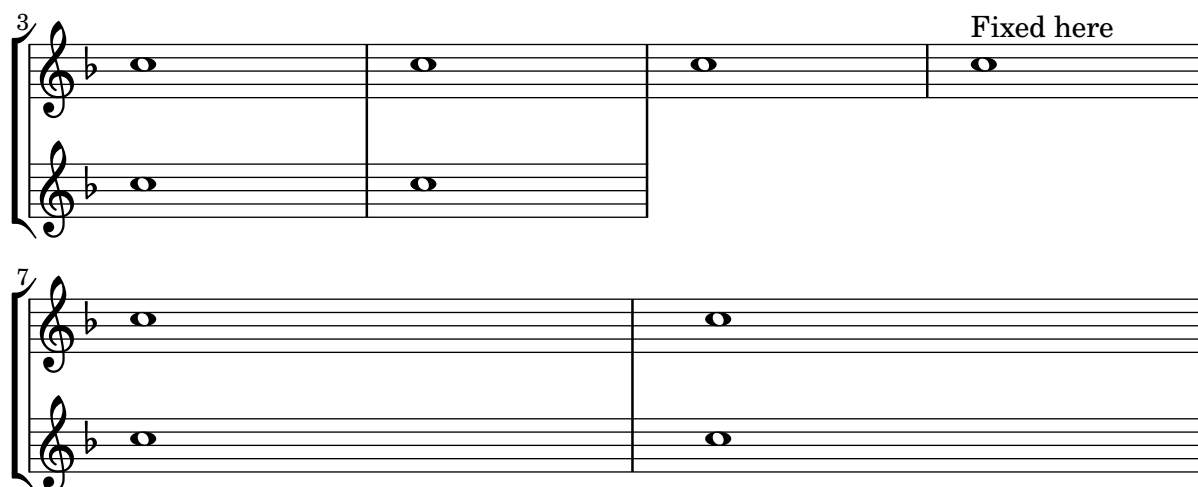


## Adding an extra staff at a line break

When adding a new staff at a line break, some extra space is unfortunately added at the end of the line before the break (to fit in a key signature change, which will never be printed anyway). The workaround is to set the `explicitKeySignatureVisibility` property of the `Staff` grob as is shown in the example.

```
\score {
  \new StaffGroup \relative c'' {
    \new Staff
    \key f \major
    c1 c^"Unwanted extra space" \break
    << { c1 | c }
    \new Staff {
      \key f \major
      \once \omit Staff.TimeSignature
      c1 | c
    }
  }
  >>
  c1 | c^"Fixed here" \break
  << { c1 | c }
  \new Staff {
    \once \set Staff.explicitKeySignatureVisibility =
      #end-of-line-invisible
    \key f \major
    \once \omit Staff.TimeSignature
    c1 | c
  }
  >>
}
}
```





## Adding indicators to staves which get split after a break

This snippet defines the commands `\splitStaffBarLine`, `\convUpStaffBarLine`, and `\convDownStaffBarLine`. These add arrows at a bar line to denote that several voices sharing a staff will each continue on a staff of their own in the next system, or that voices split in this way recombine.

Note that the implementation in this snippet draws dimensionless arrows into the right margin. For normal printing, this doesn't cause problems. However, it is necessary to increase the bounding box horizontally if you render the code as an image to avoid cropping, as demonstrated below.

```
#(define-markup-command (arrow-at-angle layout props angle-deg length fill)
  (number? number? boolean?)
  (let* ((PI-OVER-180 (/ (atan 1 1) 34))
        (degrees->radians (lambda (degrees) (* degrees PI-OVER-180)))
        (angle-rad (degrees->radians angle-deg))
        (target-x (* length (cos angle-rad)))
        (target-y (* length (sin angle-rad))))
    (interpret-markup layout props
      (markup
        #:translate (cons (/ target-x 2) (/ target-y 2))
        #:rotate angle-deg
        #:translate (cons (/ length -2) 0)
        #:concat (#:draw-line (cons length 0)
          #:arrow-head X RIGHT fill))))))

splitStaffBarLineMarkup = \markup \with-dimensions #'(0 . 0) #'(0 . 0) {
  \combine
  \arrow-at-angle #45 #(sqrt 8) ##t
  \arrow-at-angle #-45 #(sqrt 8) ##t
}

splitStaffBarLine = {
  \once \override Staff.BarLine.stencil =
  #(lambda (grob)
    (ly:stencil-combine-at-edge
      (ly:bar-line::print grob)
      X RIGHT
```

```

        (grob-interpret-markup grob splitStaffBarLineMarkup)
    0))
\break
}

convDownStaffBarLine = {
  \once \override Staff.BarLine.stencil =
  #(lambda (grob)
    (ly:stencil-combine-at-edge
      (ly:bar-line::print grob)
      X RIGHT
      (grob-interpret-markup grob #{
        \markup\with-dimensions #'(0 . 0) #'(0 . 0) {
          \translate #'(0 . -.13)\arrow-at-angle #-45 #(\sqrt 8) ##t
        }#})
    0))
\break
}

convUpStaffBarLine = {
  \once \override Staff.BarLine.stencil =
  #(lambda (grob)
    (ly:stencil-combine-at-edge
      (ly:bar-line::print grob)
      X RIGHT
      (grob-interpret-markup grob #{
        \markup\with-dimensions #'(0 . 0) #'(0 . 0) {
          \translate #'(0 . .14)\arrow-at-angle #45 #(\sqrt 8) ##t
        }#})
    0))
\break
}

\paper {
  indent = 10\mm
  short-indent = 10\mm
  line-width = 8\cm
}

separateSopranos = {
  \set Staff.instrumentName = "AI AII"
  \set Staff.shortInstrumentName = "AI AII"
  \splitStaffBarLine
  \change Staff = "up"
}

convSopranos = {
  \convDownStaffBarLine
  \change Staff = "shared"
  \set Staff.instrumentName = "S A"
  \set Staff.shortInstrumentName = "S A"
}

```

```

sI = {
  \voiceOne
  \repeat unfold 4 f''2
  \separateSopranos
  \repeat unfold 4 g''2
  \convSopranos
  \repeat unfold 4 c''2
}
sII = {
  s1*2
  \voiceTwo
  \change Staff = "up"
  \repeat unfold 4 d''2
}
aI = {
  \voiceTwo
  \repeat unfold 4 a'2
  \voiceOne
  \repeat unfold 4 b'2
  \convUpStaffBarLine
  \voiceTwo
  \repeat unfold 4 g'2
}
aII = {
  s1*2
  \voiceTwo
  \repeat unfold 4 g'2
}
ten = {
  \voiceOne
  \repeat unfold 4 c'2
  \repeat unfold 4 d'2
  \repeat unfold 4 c'2
}
bas = {
  \voiceTwo
  \repeat unfold 4 f2
  \repeat unfold 4 g2
  \repeat unfold 4 c2
}

\markup \pad-x #3 % avoid cropping
\score {
  <<
  \new ChoirStaff <<
  \new Staff = up \with {
    instrumentName = "SI SII"
    shortInstrumentName = "SI SII"
  } {
    s1*4
  }
}

```

```

\new Staff = shared \with {
  instrumentName = "S A"
  shortInstrumentName = "S A"
} <<
  \new Voice = sopI \sI
  \new Voice = sopII \sII
  \new Voice = altI \aI
  \new Voice = altII \aII
>>
\new Lyrics \with {
  alignBelowContext = up
}
\lyricsto sopII { e f g h }
\new Lyrics \lyricsto altI { a b c d e f g h i j k l }

\new Staff = men \with {
  instrumentName = "T B"
  shortInstrumentName = "T B"
} <<
  \clef F
  \new Voice = ten \ten
  \new Voice = bas \bas
>>
\new Lyrics \lyricsto bas { a b c d e f g h i j k l }
>>
>>

\layout {
  \context {
    \Staff \RemoveEmptyStaves
    \override VerticalAxisGroup.remove-first = ##t
  }
}

```

The image displays three musical staves with lyrics. The first staff shows Soprano (S A) and Tenor/Bass (T B) parts with lyrics 'a b c d'. The second staff shows Soprano I (SI SII) and Alto I (AI AII) parts with lyrics 'e f g h'. The third staff shows Soprano (S A) and Tenor/Bass (T B) parts with lyrics 'i j k l'.

## Adding orchestral cues to a vocal score

This snippet shows one approach to simplify adding many orchestral cues to the piano reduction in a vocal score. The music function `\cueWhile` takes four arguments: the music from which the cue is to be taken, as defined by `\addQuote`, the name to be inserted before the cue notes, then either UP or DOWN to specify either `\voiceOne` with the name above the staff or `\voiceTwo` with the name below the staff, and finally the piano music in parallel with which the cue notes are to appear. The name of the cued instrument is positioned to the left of the cued notes. Many passages can be cued, but they cannot overlap each other in time.

```
cueWhile =
#(define-music-function
  (instrument name dir music)
  (string? string? ly:dir? ly:music?)
  #{
    \cueDuring $instrument #dir {
      \once \override TextScript.self-alignment-X = #RIGHT
      \once \override TextScript.direction = $dir
```

```

        <>-\markup { \tiny #name }
    $music
}
#})

flute = \relative c'' {
    \transposition c'
    s4 s4 e g
}
\addQuote "flute" { \flute }

clarinet = \relative c' {
    \transposition bes
    fis4 d d c
}
\addQuote "clarinet" { \clarinet }

singer = \relative c'' { c4. g8 g4 bes4 }
words = \lyricmode { here's the lyr -- ics }

pianoRH = \relative c'' {
    \transposition c'
    \cueWhile "clarinet" "Clar." #DOWN { c4. g8 }
    \cueWhile "flute" "Flute" #UP { g4 bes4 }
}
pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  <<
    \new Staff {
      \new Voice = "singer" {
        \singer
      }
    }
    \new Lyrics {
      \lyricsto "singer"
      \words
    }
    \new PianoStaff <<
      \new Staff {
        \new Voice {
          \pianoRH
        }
      }
      \new Staff {
        \clef "bass"
        \pianoLH
      }
    >>
  >>
}

```



## Alternative bar numbering

Setting the `alternativeNumberingStyle` context property, two additional methods are available for enumerating bar numbers in repeats.

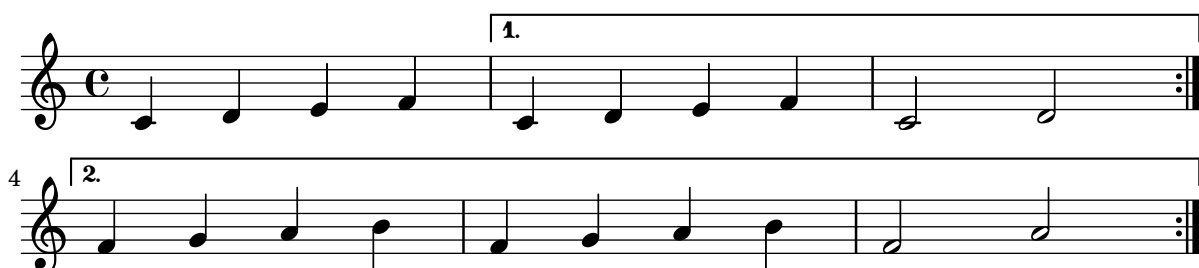
```
music = \relative c' {
  \repeat volta 3 {
    c4 d e f |
    \alternative {
      \volta 1 { c4 d e f | c2 d \break }
      \volta 2 { f4 g a b | f4 g a b | f2 a | \break }
      \volta 3 { c4 d e f | c2 d } } }
  c1 \bar "|."
}

\markup "default"
{
  \music
}

\markup \typewriter "numbers"
{
  \set Score.alternativeNumberingStyle = #'numbers
  \music
}

\markup \typewriter "numbers-with-letters"
{
  \set Score.alternativeNumberingStyle = #'numbers-with-letters
  \music
}
```

default



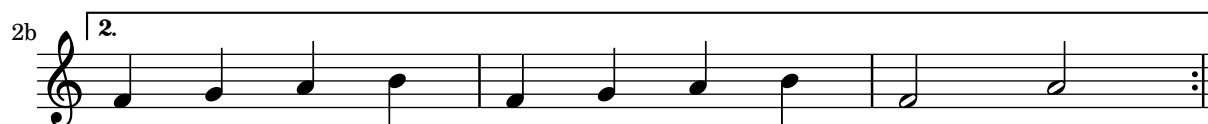
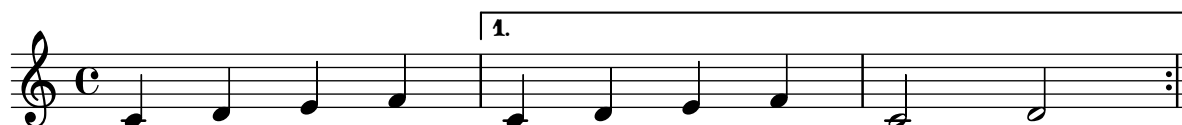




numbers



numbers-with-letters



## Ambitus after key signature

By default, ambitus are positioned at the left of the clef. The `\ambitusAfter` function allows for changing this placement. Syntax is `\ambitusAfter grob-interface`; see Graphical Object Interfaces (<http://lilypond.org/doc/v2.24/Documentation/internals/graphical-object-interfaces>) for a list of possible values for *grob-interface*.

A common use case is printing the ambitus between key signature and time signature.

```
\new Staff \with {
  \consists Ambitus_engraver
} \relative {
  \ambitusAfter key-signature
  \key d \major
  es'8 g bes cis d2
}
```

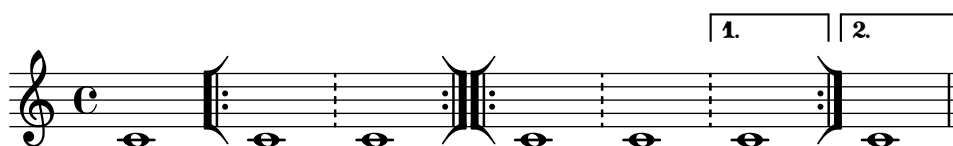


## Changing the default bar lines

Default bar lines can be changed when re-defined in a Score context.

```
\layout {
  \context {
    \Score
    % Changing the defaults from engraver-init.ly
    measureBarType = "!"
    startRepeatBarType = "[|:"
    endRepeatBarType = ":|]"
    doubleRepeatBarType = ":||[|:"
  }
}

{
  c'1
  \repeat volta 2 { c' c' }
  \repeat volta 2 { c' c' \alternative { \volta 1 { c' }
                                         \volta 2 { c' } } }
  \bar "|."
}
```



## Changing the number of lines in a staff

The number of lines in a staff may be changed by overriding the StaffSymbol property line-count.

```
upper = \relative c'' {
  c4 d e f
}

lower = \relative c {
  \clef bass
  c4 b a g
}

\score {
  \context PianoStaff <<
    \new Staff {
      \upper
    }
    \new Staff {
      \override Staff.StaffSymbol.line-count = #4
      \lower
    }
  >>
}
```



## Changing the staff size

Though the simplest way to resize staves is to use `#(set-global-staff-size size)`, an individual staff's size can be changed by scaling the properties `staff-space` and `fontSize`.

```
<<
  \new Staff {
    \relative c'' {
      \dynamicDown
      c8\ff c c c c c c c
    }
  }
  \new Staff \with {
    fontSize = #-3
    \override StaffSymbol.staff-space = #(magstep -3)
  } {
    \clef bass
    c8 c c c c\ff c c c
  }
>>
```



## Creating blank staves

To create blank staves, generate empty measures then remove the `Bar_number_engraver` from the `Score` context, and the `Time_signature_engraver`, `Clef_engraver` and `Bar_engraver` from the `Staff` context.

```
#(set-global-staff-size 10) % for the documentation
% #(set-global-staff-size 20) % for letter and A4
```

```
\book {
  \score {
    { \repeat unfold 12 { s1 \break } }

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

```

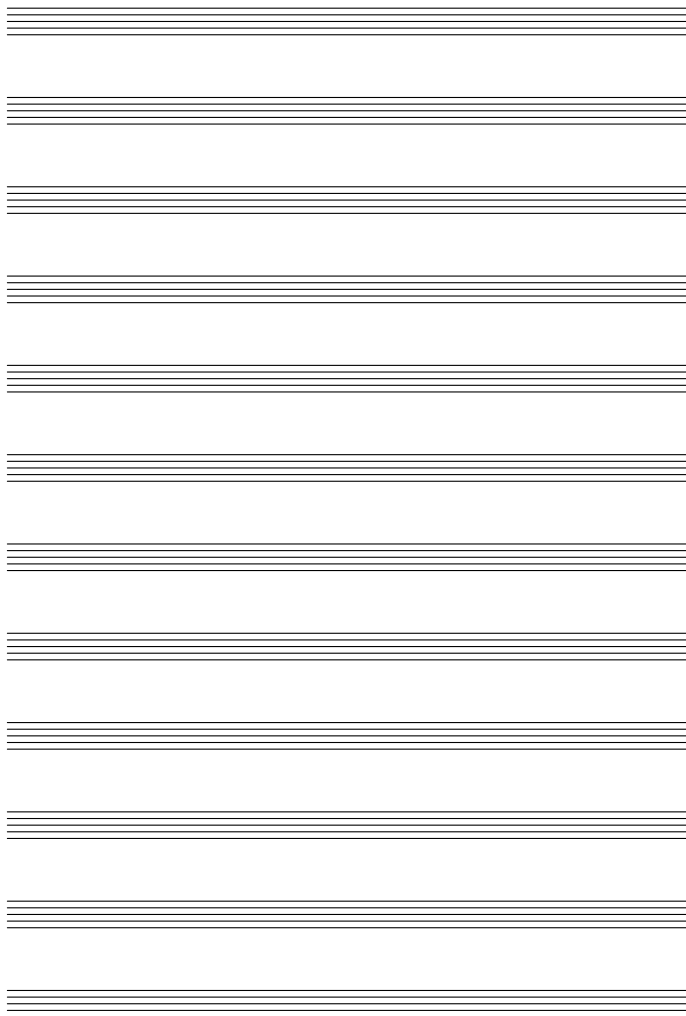
        \remove "Bar_engraver"
    }
    \context {
        \Score
        \remove "Bar_number_engraver"
    }
}

% for the documentation
\paper {
    #(set-paper-size "a6")
    ragged-last-bottom = ##f
    line-width = 90\mm
    left-margin = 7.5\mm
    bottom-margin = 5\mm
    top-margin = 5\mm
    tagline = ##f
}

% uncomment these lines for "letter" size
%{
\paper {
    #(set-paper-size "letter")
    ragged-last-bottom = ##f
    line-width = 7.5\in
    left-margin = 0.5\in
    bottom-margin = 0.25\in
    top-margin = 0.25\in
    tagline = ##f
}
}%

% uncomment these lines for "A4" size
%{
\paper {
    #(set-paper-size "a4")
    ragged-last-bottom = ##f
    line-width = 180\mm
    left-margin = 15\mm
    bottom-margin = 10\mm
    top-margin = 10\mm
    tagline = ##f
}
}%
}

```



## Creating custom key signatures

LilyPond supports custom key signatures. In this example, print for D minor and D major with an extended range of shown flats.

```
\new Staff \with {
  \override StaffSymbol.line-count = #8
  \override KeySignature.flat-positions = #'((-7 . 6))
  \override KeyCancellation.flat-positions = #'((-7 . 6))
  \override KeySignature.sharp-positions = #'((-6 . 7))
  \override KeyCancellation.sharp-positions = #'((-6 . 7))

  \override Clef.stencil =
    #(lambda (grob)
      (grob-interpret-markup grob
        #{ \markup\combine
          \musicglyph "clefs.C"
          \translate #'(-3 . -2)
          \musicglyph "clefs.F"
        #}))
      clefPosition = #3
      middleCPosition = #3
      middleCClefPosition = #3
```

```

}

{
  \key d\minor f bes, f bes, |
  \key d\major fis b, fis b, |
}

```



## Cross-staff stems

This snippet shows how to use `Span_stem_engraver` and `\crossStaff` to connect stems across staves automatically.

The stem lengths need not be specified, as the variable distance between noteheads and staves is calculated automatically. However, it is important that `\crossStaff` is applied to the correct voice or staff (i.e., on the opposite side of where a beam is or would be positioned) to get the desired effect.

```

\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
    \voiceTwo
    % Down to lower staff
    \crossStaff { e'8 e'8 } e'4 |
  }

  \new Staff {
    \clef bass
    \voiceOne
    % Up to upper staff
    \crossStaff { <e g>4 e, g16 a8. c8 } d |
    g8 f g4 \voiceTwo g8 g g4 |
  }
>>

```

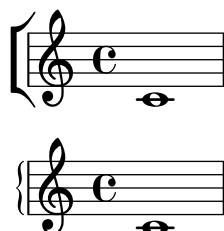


## Display bracket with only one staff in a system

If there is only one staff in a `ChoirStaff` or `StaffGroup` context, the bracket and the starting bar line will not be displayed by default. This can be changed by setting the `collapse-height` property to a value 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
    }
  >>
}
```



## Displaying a whole GrandStaff system if only one of its staves is alive

In many orchestral scores it is custom to not show staves for instruments that are silent for a while; this is called a ‘Frenched’ score. LilyPond provides this functionality via the `\RemoveEmptyStaves` command.

When they play again it is often preferred to show the staves of *all instruments of such a group*. This can be done by adding the `Keep_alive_together_engraver` to the grouping context (e.g., `GrandStaff` or `StaffGroup`).

In the example below the violins are silent in the second system. Only the first violin plays the last measure in the third system but the staff of the second violin is also displayed.

```
\score {
  <<
    \new Staff = "Staff_flute" \with {
      instrumentName = "Flute"
      shortInstrumentName = "Fl"
    }
```

```

} \relative c' {
  \repeat unfold 3 { c'4 c c c | c c c c | c c c c | \break }
}

\new StaffGroup = "StaffGroup_Strings" <<
  \new GrandStaff = "GrandStaff_violins" <<
    \new Staff = "StaffViolinI" \with {
      instrumentName = "Violin I"
      shortInstrumentName = "Vi I"
    } \relative c'' {
      a1 | R1*7 | \repeat unfold 12 a16 a4 |
    }
    \new Staff = "StaffViolinII" \with {
      instrumentName = "Violin II"
      shortInstrumentName = "Vi II"
    } \relative c' {
      e1 | R1*8 |
    }
  >>

  \new Staff = "Staff_cello" \with {
    instrumentName = "Cello"
    shortInstrumentName = "Ce"
  } \relative c {
    \clef bass \repeat unfold 9 { c1 } |
  }
  >>
  >>
}

\layout {
  indent = 3.0\cm
  short-indent = 1.5\cm

  \context {
    \GrandStaff
    \consists Keep_alive_together_engraver
  }
  \context {
    \Staff
    \RemoveEmptyStaves
  }
}

```



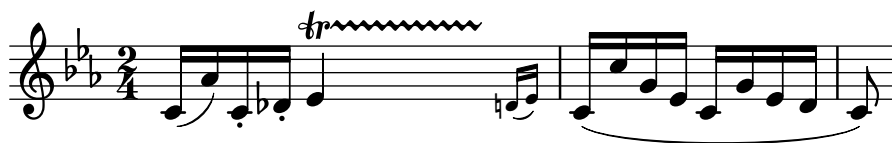
The image displays three musical staves, each representing a different instrument: Flute, Violin I, Violin II, and Cello. The first staff shows a continuous melody for the Flute. The second staff shows a trill for Violin I, Violin II, and Cello. The third staff shows a trill for Violin I, Violin II, and Cello, with a trill spanner extending across the measures.

## Extending a trill spanner

For `TrillSpanner` grobs, the `minimum-length` property becomes effective only if the `set-spacing-rods` procedure is called explicitly.

To do this, the `springs-and-rods` property should be set to `ly:spanner::set-spacing-rods`.

```
\relative c' {
  \key c\minor
  \time 2/4
  c16( as') c,-. des-.
  \once\override TrillSpanner.minimum-length = #15
  \once\override TrillSpanner.springs-and-rods = #ly:spanner::set-spacing-rods
  \afterGrace es4\startTrillSpan { d16[(\stopTrillSpan es)] }
  c( c' g es c g' es d
  c8)
}
```



## Extending glissandi across repeats

A glissando that 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 that takes the pitch of the grace note as its argument.

Note that in polyphonic music the grace note must be matched with corresponding grace notes in all other voices.

```
repeatGliss = #(define-music-function (grace)
  (ly:pitch?)
  #{
    % the next two lines ensure the glissando is long enough
    % to be visible
    \once \override Glissando.springs-and-rods
      = #ly:spanner::set-spacing-rods
    \once \override Glissando.minimum-length = 3.5
    \once \hideNotes
    \grace $grace \glissando
  #})

\score {
  \relative c'' {
    \repeat volta 3 { c4 d e f\glissando }
    \alternative {
      { g2 d }
      { \repeatGliss f g2 e }
      { \repeatGliss f e2 d }
    }
  }
}

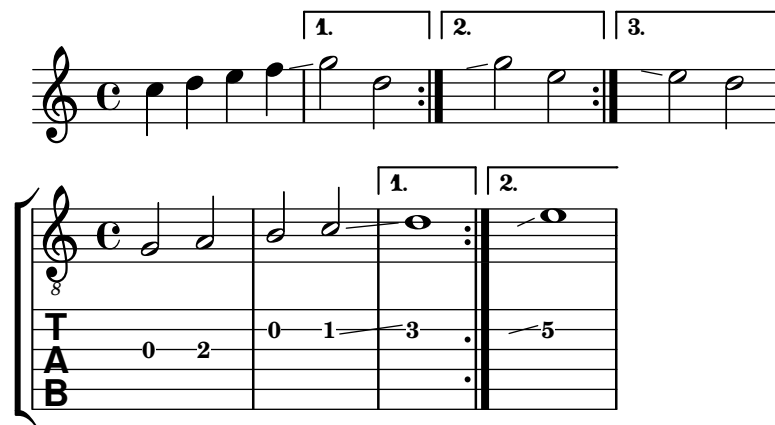
music = \relative c' {
  \voiceOne
  \repeat volta 2 {
    g a b c\glissando
  }
  \alternative {
    { d1 }
    { \repeatGliss c \once \omit StringNumber e1\2 }
  }
}

\score {
  \new StaffGroup <<
    \new Staff <<
      \new Voice { \clef "G_8" \music }
    >>
  >>
}
```

```

\new TabStaff <<
  \new TabVoice { \clef "moderntab" \music }
>>
>>
}

```



## Flat ties

This snippet provides a function `flared-tie` to draw a tie that consist of straight lines. It is intended as a replacement for the default tie-drawing function (i.e., a replacement argument for the `stencil` property of the `Tie` grob).

The argument of `flared-tie` is a list of coordinate pairs that specify additional points between the first and last point to span up the tie's lines. The first and last point are identical to the original tie's start and end point, respectively. The X and Y coordinate values are multiples of the bounding box length and height of the original tie (also taking care of the tie's direction); consequently, the first point has coordinates (0,0), and the last point (1,0).

The function `flare-tie` defines a shorthand for a flat tie. Further tweaking of the shape is possible by overriding `Tie.details.height-limit` or with `\shape`. It is also possible to change the custom definition on the fly.

```

#(define ((flared-tie coords) grob)
  (define (pair-to-list pair)
    (list (car pair) (cdr pair)))

  (define (normalize-coords goods x y dir)
    (map
      (lambda (coord)
        (cons (* x (car coord)) (* y dir (cdr coord))))
      goods))

  (define (my-c-p-s points thick)
    (make-connected-path-stencil points thick 1.0 1.0 #f #f))

  ;; Calling `ly:tie::print` and assigning its return value to a
  ;; variable in this outer `let` triggers LilyPond to position the
  ;; tie, allowing us to extract its extents. We only proceed,
  ;; however, if the tie doesn't get discarded (for whatever reason).
  (let ((sten (ly:tie::print grob)))
    (if (grob::is-live? grob)
        (let* ((layout (ly:grob-layout grob))

```

```

(line-thickness (ly:output-def-lookup layout
                'line-thickness))
(thickness (ly:grob-property grob 'thickness 0.1))
(used-thick (* line-thickness thickness))
(dir (ly:grob-property grob 'direction))
(xex (ly:stencil-extent sten X))
(yex (ly:stencil-extent sten Y))
(lenx (interval-length xex))
(leny (interval-length yex))
(xtrans (car xex))
(ytrans (if (> dir 0)(car yex) (cdr yex)))
;; Add last point.
(coord-list (append coords '((1.0 . 0.0))))
(uplist
  (map pair-to-list
    (normalize-coords coord-list lenx (* leny 2) dir))))
(ly:stencil-translate
  (my-c-p-s uplist used-thick)
  (cons xtrans ytrans)))
'()))

% Define a default tie shape consisting of three straight lines.
#(define flare-tie
  (flared-tie '((0.1 . 0.3) (0.9 . 0.3))))

\relative c' {
  a4~ a
  \once \override Tie.stencil = #flare-tie
  a4~ a \break

  <a c e a c e a c e>~ q
  \once \override Tie.stencil = #flare-tie
  q~ q\break

  <>~\markup \small \typewriter "height-limit = 14"
  \override Tie.details.height-limit = 14
  a'4~ a
  \once \override Tie.stencil = #flare-tie
  a4~ a \break

  <>~\markup \small \typewriter "height-limit = 0.5"
  \override Tie.details.height-limit = 0.5
  a4~ a
  \once \override Tie.stencil = #flare-tie
  a4~ a \break

  \revert Tie.details.height-limit

  <>~\markup \small \typewriter
    "\shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0))"
  \shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0)) Tie
  a4~ a

```

```

\once \override Tie.stencil = #flare-tie
\shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0)) Tie
a4~ a \break

<>^\markup \small \typewriter
      "#(flared-tie '((0.2 . 2) (0.5 . -3) (0.8 . 1)))"
\once \override Tie.stencil =
      #(flared-tie '((0.2 . 2) (0.5 . -3) (0.8 . 1)))
a4~ a
<>_\markup \small \typewriter
      "#(flared-tie '((0.5 . 2)))"
\once \override Tie.stencil = #(flared-tie '((0.5 . 2)))
a'4~ a
}

```

1

2

3 height-limit = 14

4 height-limit = 0.5

5 \shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0))

#(flared-tie '((0.2 . 2) (0.5 . -3) (0.8 . 1)))

6

#(flared-tie '((0.5 . 2)))

## Forcing measure width to adapt to a metronome mark's width

By default, metronome marks do not influence horizontal spacing. This can be solved through a simple override, as shown in the second half of the example.

```

example = {
  R1
  \tempo "Allegro molto" R1*6
  \tempo "poco rit." R1*2
}

```

```

\tempo "a tempo" R1*8 \break
}

{
  \compressMMRests {
    \example
    \override Score.MetronomeMark.extra-spacing-width = #'(-3 . 0)
    \example
  }
}

\layout {
  ragged-right = ##t
}

```

The image shows two staves of musical notation. The first staff starts with a treble clef and a common time signature 'C'. It contains three measures: the first has a whole rest, the second has a half rest, and the third has a whole note. Above the staff, tempo markings are placed: 'Allegro molto' above the first measure, 'poco rit.' above the second measure, and 'a tempo' above the third measure. Below the staff, the numbers 6, 2, and 8 are placed under the first, second, and third measures respectively. The second staff starts with a treble clef and a common time signature 'C'. It also contains three measures: the first has a whole rest, the second has a half rest, and the third has a whole note. Above the staff, tempo markings are placed: 'Allegro molto' above the first measure, 'poco rit.' above the second measure, and 'a tempo' above the third measure. Below the staff, the numbers 6, 2, and 8 are placed under the first, second, and third measures respectively. The number 18 is written to the left of the second staff.

## Glissandi can skip grobs

NoteColumn grobs can be skipped over by glissandi.

```

\relative c' {
  a2 \glissando
  \once \override NoteColumn.glissando-skip = ##t
  f''4 d,
}

```

The image shows a single staff of musical notation with a treble clef and a common time signature 'C'. It contains three measures: the first has a whole note, the second has a half note, and the third has a whole note. A glissando line is drawn from the first measure to the second measure, indicating a glissando effect.

## Incipit

When transcribing mensural music, an incipit at the beginning of the piece is useful to indicate the original key and tempo. While today musicians are used to bar lines in order to faster recognize rhythmic patterns, bar lines were not yet invented during the period of mensural music; in fact, the meter often changed after every few notes. As a compromise, bar lines are often printed between the staves rather than on the staves.

*% A short excerpt from the Jubilate Deo by Orlande de Lassus*

```

global = {
  \set Score.skipBars = ##t
  \key g \major
  \time 4/4
}

```

```

% the actual music
\skip 1*8

% let finis bar go through all staves
\override Staff.BarLine.transparent = ##f

% finis bar
\bar "|."
}

discantusIncipit = \new PetrucciStaff {
  \clef "petrucci-c1"
  \key f \major
  \time 2/2
  c'1.
}

discantusNotes = {
  \transpose c' c'' {
    \clef "treble"
    d'2. d'4 |
    b e' d'2 |
    c'4 e'4.( d'8 c' b |
    a4) b a2 |
    b4.( c'8 d'4) c'4 |
    \once \hide NoteHead
    c'1 |
    b\breve |
  }
}

discantusLyrics = \lyricmode {
  Ju -- bi -- la -- te De -- o,
  om -- nis ter -- ra, __ om-
  "...
  -us.
}

altusIncipit = \new PetrucciStaff {
  \clef "petrucci-c3"
  \key f \major
  \time 2/2
  e'1\rest f'1.
}

altusNotes = {
  \transpose c' c'' {
    \clef "treble"
    r2 g2. e4 fis g |
    a2 g4 e |
    fis g4.( fis16 e fis4) |
    g1 |
  }
}

```

```

    \once \hide NoteHead
    g1 |
    g\breve |
  }
}

altusLyrics = \lyricmode {
  Ju -- bi -- la -- te
  De -- o, om -- nis ter -- ra,
  "...
  -us.
}

tenorIncipit = \new PetrucciStaff {
  \clef "petrucci-c4"
  \key f \major
  \time 2/2
  r\longa
  r\breve
  r1 c'1.
}

tenorNotes = {
  \transpose c' c' {
    \clef "treble_8"
    R1 |
    R1 |
    R1 |
    % two measures
    r2 d'2. d'4 b e' |
    \once \hide NoteHead
    e'1 |
    d'\breve |
  }
}

tenorLyrics = \lyricmode {
  Ju -- bi -- la -- te
  "...
  -us.
}

bassusIncipit = \new PetrucciStaff {
  % The original print shows the b flat
  % for the f major key signature twice.
  \override Staff.KeySignature.flat-positions = #'((-7 . 6))
  \clef "mensural-f"
  \key f\major
  \time 2/2
  \tweak Y-offset #1 r\longa \tweak Y-offset #1 r\longa
  f1.
}

```



```

bassusNotes = {
  \transpose c' c' {
    \clef "bass"
    R1 |
    R1 |
    R1 |
    R1 |
    g2. e4 |
    \once \hide NoteHead
    e1 |
    g\breve |
  }
}

bassusLyrics = \lyricmode {
  Ju -- bi-
  "...
  -us.
}

\score {
  <<
  \new StaffGroup = choirStaff <<
    \new Voice = "discantusNotes" <<
      \set Staff.instrumentName = "Discantus"
      \incipit #1 \discantusIncipit
      \global
      \discantusNotes
    >>
    \new Lyrics \lyricsto discantusNotes { \discantusLyrics }
    \new Voice = "altusNotes" <<
      \set Staff.instrumentName = "Altus"
      \global
      \incipit #1 \altusIncipit
      \altusNotes
    >>
    \new Lyrics \lyricsto altusNotes { \altusLyrics }
    \new Voice = "tenorNotes" <<
      \set Staff.instrumentName = "Tenor"
      \global
      \incipit #1 \tenorIncipit
      \tenorNotes
    >>
    \new Lyrics \lyricsto tenorNotes { \tenorLyrics }
    \new Voice = "bassusNotes" <<
      \set Staff.instrumentName = "Bassus"
      \global
      \incipit #1 \bassusIncipit
      \bassusNotes
    >>
    \new Lyrics \lyricsto bassusNotes { \bassusLyrics }
  >>
}

```

```

>>
>>
\layout {
  \context {
    \Score
    %% no bar lines in staves or lyrics
    \hide BarLine
  }
  %% the next two instructions keep the lyrics between the bar lines
  \context {
    \Lyrics
    \consists "Bar_engraver"
    \consists "Separating_line_group_engraver"
  }
  \context {
    \Voice
    %% no slurs
    \hide Slur
    %% Comment in the below "\remove" command to allow line
    %% breaking also at those bar lines where a note overlaps
    %% into the next measure. The command is commented out in this
    %% short example score, but especially for large scores, you
    %% will typically yield better line breaking and thus improve
    %% overall spacing if you comment in the following command.
    %%\remove "Forbid_line_break_engraver"
  }
  indent = 5\cm
  incipit-width = 2.5\cm
}
}

```

Discantus

Altus

Tenor

Bassus

Ju - bi - la - te De - o, om -

Ju - bi - la - te De - o, om -

8

## Inserting score fragments above a staff, as markups

The `\markup` command is quite versatile. In this snippet, it contains a `\score` block instead of texts or marks.

```
tuning = \markup \score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \clef bass
    <c, g, d g>1
  }
  \layout {
    indent = 0\cm
  }
}

\header {
  title = "Solo Cello Suites"
  subtitle = "Suite IV"
  subsubtitle = \markup { Originalstimmung: \raise #0.5 \tuning }
  tagline = ##f
}

\layout {
  ragged-right = ##f
}

\relative c' {
  \time 4/8
  \tuplet 3/2 { c8 d e } \tuplet 3/2 { c d e }
  \tuplet 3/2 { c8 d e } \tuplet 3/2 { c d e }
  g8 a g a
  g8 a g a
}
```

## Solo Cello Suites

### Suite IV

Originalstimmung: 



### Let TabStaff print the topmost string at bottom

In tablatures, the first string is usually printed topmost. If you want to have it at the bottom, set the `stringOneTopmost` context property to `##f`. For a context-wide setting this could be done in the `\layout` block as well.

```
%\layout {
% \context {
%   \Score
%   stringOneTopmost = ##f
% }
% \context {
%   \TabStaff
%   tablatureFormat = #fret-letter-tablature-format
% }
%}

m = {
  \cadenzaOn
  e, b, e gis! b e'
  \bar "||"
}

<<
  \new Staff {
    \clef "G_8"
    <>_"default" \m
    <>_"italian (historic)"\m
  }
  \new TabStaff
  {
    \m
    \set Score.stringOneTopmost = ##f
    \set TabStaff.tablatureFormat = #fret-letter-tablature-format
    \m
  }
>>
```

## Letter tablature formatting

Tablature can be formatted using letters instead of numbers.

```
music = \relative c {
  c4 d e f
  g4 a b c
  d4 e f g
}
```

```
<<
  \new Staff {
    \clef "G_8"
    \music
  }
  \new TabStaff \with {
    tablatureFormat = #fret-letter-tablature-format
  } {
    \music
  }
>>
```

The first system of the musical score for 'The Rose Tree' is shown. It features a treble clef, a common time signature (C), and a key signature of one flat (B-flat). The melody is written on a single staff. Below the staff, the lyrics 'The Rose Tree' are written in a stylized font. The melody consists of a series of eighth and quarter notes, starting on a B-flat and ascending to a D.

## Making glissandi breakable

Normally, LilyPond refuses to automatically break a line at places where a glissando crosses a bar line. This behavior can be changed by setting the `Glissando.breakable` property to `#t`. Also setting the `after-line-breaking` property to `#t` makes the glissando line continue after the break.

The breakable property does not affect manual breaks inserted with commands like `\break`.

```
glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}
```

```
music = {
```

```

\repeat unfold 16 f8 |
f1\glissando |
a4 r2. |
\repeat unfold 16 f8 |
f1\glissando \once\glissandoSkipOn |
a2 a4 r4 |
\repeat unfold 16 f8
}

\relative c'' {
  <>^\markup { \typewriter Glissando.breakable
               set to \typewriter "#t" }
  \override Glissando.breakable = ##t
  \override Glissando.after-line-breaking = ##t
  \music
}

\relative c'' {
  <>^\markup { \typewriter Glissando.breakable not set }
  \music
}

\paper {
  line-width = 100\mm
}

```



## 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'
}
```



## Measure counters

This snippet demonstrates the use of the `Measure_counter_engraver` to number groups of successive measures. Any stretch of measures may be numbered, whether consisting of repetitions or not.

The engraver must be added to the appropriate context. Here, a `Staff` context is used; another possibility is a `Dynamics` context.

The counter is begun with `\startMeasureCount` and ended with `\stopMeasureCount`. Numbering will start by default with 1, but this behavior may be modified by overriding the `count-from` property.

When a measure extends across a line break, the number will appear twice, the second time in parentheses.

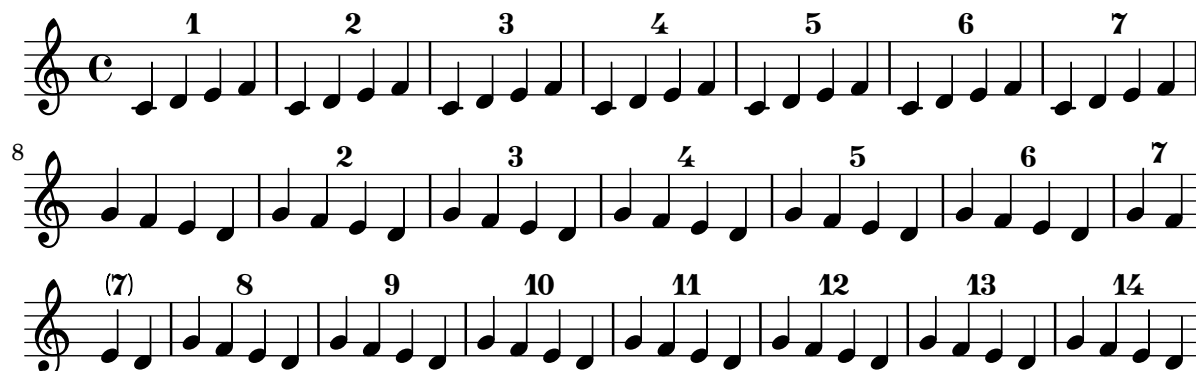
```
\layout {
  \context {
    \Staff
    \consists #Measure_counter_engraver
  }
}

\new Staff {
  \startMeasureCount
  \repeat unfold 7 {
    c'4 d' e' f'
  }
  \stopMeasureCount
  \bar "||"
  g'4 f' e' d'
  \override Staff.MeasureCounter.count-from = #2
  \startMeasureCount
  \repeat unfold 5 {
    g'4 f' e' d'
  }
  g'4 f'
  \bar ""
  \break
  e'4 d'
  \repeat unfold 7 {
```

```

    g'4 f' e' d'
  }
  \stopMeasureCount
}

```



## Mensurstriche layout (bar lines between the staves)

*Mensurstriche*, bar lines between but not through staves, can be printed by setting `measureBarType` to `"-span|"` and using a grouping context that allows span bars, such as `StaffGroup`.

```

\layout {
  \context {
    \Staff
    measureBarType = "-span|"
  }
}

```

```

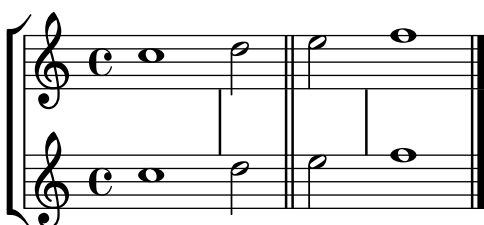
music = \fixed c'' {
  c1
  d2 \section e2
  f1 \fine
}

```

```

\new StaffGroup <<
  \new Staff \music
  \new Staff \music
>>

```



## Modifying the ottava spanner slope

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

```

\relative c'' {
  \override Staff.OttavaBracket.stencil = #ly:line-spanner::print
  \override Staff.OttavaBracket.bound-details =

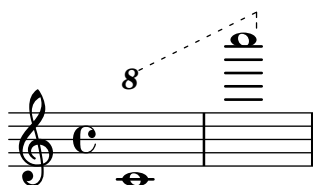
```



```

#`((left . ((Y . 0)
            (attach-dir . ,LEFT)
            (padding . 0)
            (stencil-align-dir-y . ,CENTER)))
   (right . ((Y . 5.0) ; Change the number here
            (padding . 0)
            (attach-dir . ,RIGHT)
            (text . ,(make-draw-dashed-line-markup
                      (cons 0 -1.2)))))
\override Staff.OttavaBracket.left-bound-info =
  #ly:horizontal-line-spanner::calc-left-bound-info-and-text
\override Staff.OttavaBracket.right-bound-info =
  #ly:horizontal-line-spanner::calc-right-bound-info
\ottava 1
c1
c'''1
}

```



## Nesting staves

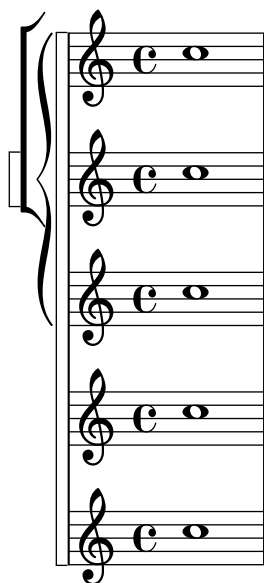
The property `systemStartDelimiterHierarchy` can be used to make more complex nested staff groups. The `systemStartDelimiterHierarchy` property of the `StaffGroup` context takes an alphabetical list of the number of staves produced. Before each staff a system start delimiter can be given. It has to be enclosed in brackets and takes as much staves as the brackets enclose. Elements in the list can be omitted, but the first bracket takes always the complete number of staves. The possibilities are `SystemStartBar`, `SystemStartBracket`, `SystemStartBrace`, and `SystemStartSquare`.

```

\new StaffGroup
\relative c'' <<
  \override StaffGroup.SystemStartSquare.collapse-height = 4
  \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 }
>>

```



## Non-traditional key signatures

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

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

```
\set Staff.keyAlterations =
  #`((octave . step) . alter) ((octave . step) . alter) ...)
```

where, for each element in the list, *octave* specifies the octave (0 being the octave from middle C to the B above), *step* specifies the note within the octave (0 means C and 6 means B), and *alter* is one of SHARP, FLAT, DOUBLE-SHARP, etc., preceded by a comma.

Alternatively, you can use the more concise format (*step . alter*) for each item in the list if the same alterations are used 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.

```
\include "arabic.ly"
```

```
\relative do' {
  \set Staff.keyAlterations = #`((0 . ,SEMI-FLAT)
                                (1 . ,SEMI-FLAT)
                                (2 . ,FLAT)
                                (5 . ,FLAT)
                                (6 . ,SEMI-FLAT))

  % \set Staff.extraNatural = ##f
  re reb \down reb resd
  dod dob dosd \down dob |
  dobsb dods do do |
}
```



## Orchestra, choir and piano template

This template demonstrates the use of nested `StaffGroup` and `GrandStaff` contexts to subgroup instruments of the same type together, and a way to use `\transpose` so that variables hold music for transposing instruments at concert pitch.

```

#(set-global-staff-size 17)

\paper {
  indent = 3.0\cm % add space for instrumentName
  short-indent = 1.5\cm % add less space for shortInstrumentName
}

fluteMusic = \relative c' { \key g \major g'1 b }

% Pitches as written on a manuscript for Clarinet in A
% are transposed to concert pitch.
clarinetMusic = \transpose c' a
  \relative c'' { \key bes \major bes1 d }

trumpetMusic = \relative c { \key g \major g''1 b }

% Key signature is often omitted for horns
hornMusic = \transpose c' f
  \relative c { d'1 fis }

percussionMusic = \relative c { \key g \major g1 b }

sopranoMusic = \relative c'' { \key g \major g'1 b }
sopranoLyrics = \lyricmode { Lyr -- ics }

altoIMusic = \relative c' { \key g \major g'1 b }
altoILyrics = \sopranoLyrics
altoIIMusic = \relative c' { \key g \major g'1 b }
altoIILyrics = \lyricmode { Ah -- ah }

tenorMusic = \relative c' { \clef "treble_8" \key g \major g1 b }
tenorLyrics = \sopranoLyrics

pianoRHMus = \relative c { \key g \major g''1 b }
pianoLHMus = \relative c { \clef bass \key g \major g1 b }

violinIMusic = \relative c' { \key g \major g'1 b }
violinIIMusic = \relative c' { \key g \major g'1 b }

violaMusic = \relative c { \clef alto \key g \major g'1 b }

celloMusic = \relative c { \clef bass \key g \major g1 b }

bassMusic = \relative c { \clef "bass_8" \key g \major g,1 b }

\book {
  \score {

```

```

<<
\new StaffGroup = "StaffGroup_woodwinds" <<
  \new Staff = "Staff_flute" \with { instrumentName = "Flute" }
    \fluteMusic

  \new Staff = "Staff_clarinet" \with {
    instrumentName = \markup { \concat { "Clarinet in B" \flat } }
  }
  % Declare that written Middle C in the music
  % to follow sounds a concert B flat, for
  % output using sounded pitches such as MIDI.
  %\transposition bes

  % Print music for a B-flat clarinet
  \transpose bes c' \clarinetMusic
>>

\new StaffGroup = "StaffGroup_brass" <<
  \new Staff = "Staff_hornI" \with {
    instrumentName = "Horn in F"
  }
  % \transposition f
  \transpose f c' \hornMusic

  \new Staff = "Staff_trumpet" \with {
    instrumentName = "Trumpet in C"
  }
  \trumpetMusic
>>

\new RhythmicStaff = "RhythmicStaff_percussion" \with {
  instrumentName = "Percussion"
}
\percussionMusic

\new PianoStaff \with {
  instrumentName = "Piano"
} <<
  \new Staff { \pianoRHMusical }
  \new Staff { \pianoLHMusical }
>>

\new ChoirStaff = "ChoirStaff_choir" <<
  \new Staff = "Staff_soprano" \with {
    instrumentName = "Soprano"
  }
  \new Voice = "soprano" \sopranoMusical
  \new Lyrics \lyricsto "soprano" { \sopranoLyrics }

  \new GrandStaff = "GrandStaff_alto" \with {
    \accepts Lyrics
  } <<

```

```

\new Staff = "Staff_altoI" \with {
  instrumentName = "Alto I"
}
\new Voice = "altoI"
\altoIMusic
\new Lyrics \lyricsto "altoI" { \altoILyrics }
\new Staff = "Staff_altoII" \with {
  instrumentName = "Alto II"
}
\new Voice = "altoII"
\altoIIMusic
\new Lyrics \lyricsto "altoII" { \altoIILyrics }
>>

\new Staff = "Staff_tenor" \with {
  instrumentName = "Tenor"
}
\new Voice = "tenor" \tenorMusic
\new Lyrics \lyricsto "tenor" { \tenorLyrics }
>>

\new StaffGroup = "StaffGroup_strings" <<
\new GrandStaff = "GrandStaff_violins" <<
\new Staff = "Staff_violinI" \with {
  instrumentName = "Violin I"
}
\violinIMusic
\new Staff = "Staff_violinII" \with {
  instrumentName = "Violin II"
}
\violinIIMusic
>>

\new Staff = "Staff_viola" \with {
  instrumentName = "Viola"
}
\violaMusic

\new Staff = "Staff_cello" \with {
  instrumentName = "Cello"
}
\celloMusic

\new Staff = "Staff_bass" \with {
  instrumentName = "Double Bass"
}
\bassMusic
>>
>>
}
}

```

Flute

Clarinet in B $\flat$

Horn in F

Trumpet in C

Percussion

Piano

Soprano

Alto I

Alto II

Tenor

Violin I

Violin II

Viola

Cello

Double Bass

Lyr - ics

Lyr - ics

Ah - ah

Lyr - ics

8

8

## Print chord names with same root and different bass as slash and bass note

To print subsequent ChordNames only differing in its bass note as slash and bass note, use the Scheme engraver defined in this snippet. The behaviour may be controlled in detail by the chordChanges context property.

```
#(define Bass_changes_equal_root_engraver
  (lambda (ctx)
    "For sequential `ChordNames` with the same root but a different bass,
    the root markup is dropped: D D/C D/B -> D /C /B.
    The behaviour may be controlled by setting the `chordChanges` context
    property."
    (let ((chord-pitches '())
          (last-chord-pitches '())
          (bass-pitch #f))
      (make-engraver
        ((initialize this-engraver)
         (let ((chord-note-namer (ly:context-property ctx
                                                    'chordNoteNamer)))
           ;; Set 'chordNoteNamer, respect user setting if already done
           (ly:context-set-property! ctx 'chordNoteNamer
                                     (if (procedure? chord-note-namer)
                                         chord-note-namer
                                         note-name->markup))))
          (listeners
           ((note-event this-engraver event)
            (let* ((pitch (ly:event-property event 'pitch))
                   (pitch-name (ly:pitch-notename pitch))
                   (pitch-alt (ly:pitch-alteration pitch))
                   (bass (ly:event-property event 'bass #f))
                   (inversion (ly:event-property event 'inversion #f)))
              ;; Collect notes of the chord
              ;; - to compare inversed chords we need to collect the
              ;;   bass note as usual member of the chord, whereas an
              ;;   added bass must be treated separate from the usual
              ;;   chord-notes
              ;; - notes are stored as pairs containing their
              ;;   pitch-name (an integer), i.e. disregarding their
              ;;   octave and their alteration
              (cond (bass (set! bass-pitch pitch))
                    (inversion
                     (set! bass-pitch pitch)
                     (set! chord-pitches
                           (cons (cons pitch-name pitch-alt)
                                chord-pitches)))
                    (else
                     (set! chord-pitches
                           (cons (cons pitch-name pitch-alt)
                                chord-pitches)))))))
          (acknowledgers
           ((chord-name-interface this-engraver grob source-engraver)
```

```

(let ((chord-changes (ly:context-property ctx
                                     'chordChanges #f)))
  ;; If subsequent chords are equal apart from their bass,
  ;; reset the 'text-property.
  ;; Equality is done by comparing the sorted lists of this
  ;; chord's elements and the previous chord. Sorting is
  ;; needed because inverted chords may have a different
  ;; order of pitches. `chord-changes` needs to be true.
  (if (and bass-pitch
            chord-changes
            (equal?
              (sort chord-pitches car<)
              (sort last-chord-pitches car<)))
      (ly:grob-set-property!
        grob 'text
        (make-line-markup
          (list
            (ly:context-property ctx 'slashChordSeparator)
            ((ly:context-property ctx 'chordNoteNamer)
              bass-pitch
              (ly:context-property ctx
                                    'chordNameLowercaseMinor))))))
      (set! last-chord-pitches chord-pitches)
      (set! chord-pitches '())
      (set! bass-pitch #f))))

((finalize this-engraver)
 (set! last-chord-pitches '()))))

myChords = \chordmode {
  % \germanChords

  \set chordChanges = ##t
  d2:m d:m/cis

  d:m/c
  \set chordChanges = ##f
  d:m/b

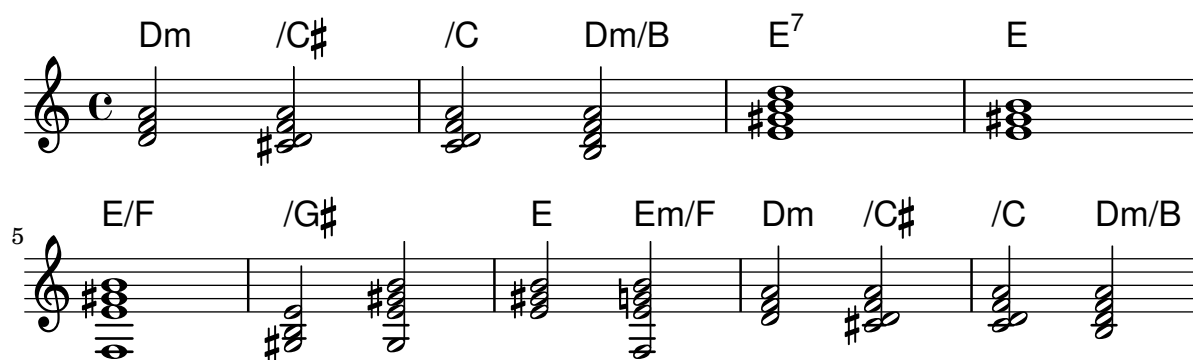
  e1:7
  \set chordChanges = ##t
  e
  \break

  \once \set chordChanges = ##f
  e1/f
  e2/gis e/+gis e e:m/f d:m d:m/cis d:m/c
  \set chordChanges = ##f
  d:m/b
}

```



```
<<
\new ChordNames
  \with { \consists #Bass_changes_equal_root_engraver }
  \myChords
\new Staff \myChords
>>
```



## Putting lyrics inside the staff

Lyrics can be moved vertically to place them inside the staff. The lyrics are moved with `\override LyricText.extra-offset = #'(0 . dy)`, and there are similar commands to move the extenders and hyphens. A good value for *dy* must be found by trial and error.

```
<<
\new Staff <<
  \new Voice = "voc" \relative c' { \stemDown a bes c8 b c4 }
>>
\new Lyrics \with {
  \override LyricText.extra-offset = #'(0 . 8.6)
  \override LyricExtender.extra-offset = #'(0 . 8.6)
  \override LyricHyphen.extra-offset = #'(0 . 8.6)
} \lyricsto "voc" { La la -- la _ _ la }
>>
```



## Quoting another voice

The `quotedEventTypes` context property determines which music event types should be quoted. The default value is `(note-event rest-event tie-event beam-event tuplet-span-event)`, which means that only notes, rests, ties, beams, and tuplets of a quoted voice appear in the `\quoteDuring` expression.

In the following example, a 16th rest is not quoted since `rest-event` is not in the redefined value of `quotedEventTypes`.

For a list of event types, consult the “Music classes” section of the Internals Reference.

```
quoteMe = \relative c' {
  fis4 r16 a8.-> b4\ff c
}
```

```

\addQuote quoteMe \quoteMe

original = \relative c'' {
  c8 d s2
  \once \override NoteColumn.ignore-collision = ##t
  es8 gis8
}

<<
  \new Staff \with { instrumentName = "quoteMe" }
  \quoteMe

  \new Staff \with { instrumentName = "orig" }
  \original

  \new Staff \with {
    instrumentName = "orig+quote"
    quotedEventTypes = #'(note-event articulation-event)
  }
  \relative c''
  <<
    \original
    \new Voice {
      s4
      \set fontSize = #-4
      \override Stem.length-fraction = #(magstep -4)
      \quoteDuring "quoteMe" { \skip 2. }
    }
  >>
>>

```

## Quoting another voice with transposition

Quotations take into account the transposition of both source and target. In this example, all instruments play sounding middle c; the target is an instrument in f. The target part may be transposed using `\transpose`. In this case, all the pitches (including the quoted ones) are transposed.

```

\addQuote clarinet {
  \transpose bes
  \repeat unfold 8 { d'16 d' d'8 }
}

```

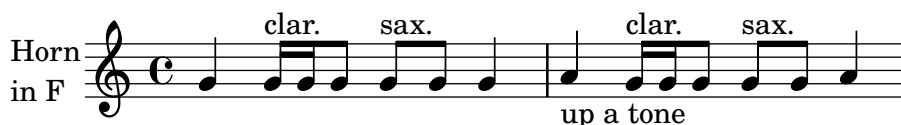
```

\addQuote sax {
  \transposition es'
  \repeat unfold 16 { a8 }
}

quoteTest = {
  % french horn
  \transposition f
  g'4
  << \quoteDuring "clarinet" { \skip 4 } s4^"clar." >>
  << \quoteDuring "sax" { \skip 4 } s4^"sax." >>
  g'4
}

{
  \new Staff \with {
    instrumentName = \markup { \column { Horn "in F" } }
  }
  \quoteTest
  \transpose c' d' << \quoteTest s4_"up a tone" >>
}

```



## Reconciling bar lines thickness when staves are of different sizes (TODO duplicates snippet in Documentation/en/notation/spacing.itely)

A regularly occurring problem in ensemble or chamber music scores, particularly with piano, is that when some of the staves are printed in a smaller size, their bar lines do not match those of larger staves. This may become particularly egregious for closing double bar lines, as demonstrated here.

There are different solutions, which all involve correcting some of the properties overridden by the `\magnifyStaff` command: namely, `kern` `thick-thickness` and `hair-thickness` (the latter of which also governs ordinary bar lines). These are properties of the `BarLine` graphical object (“*grob*”), which resides in the `Staff` context.

This can be done by using either the `\revert` command, or more advanced commands such as `revert-props`. Alternatively, some effects of `\magnifyStaff` may be replicated on other staves through the `scale-props` command, which may be used to harmonize all bar lines either to their appearance on the smallest staves, or to some intermediate value.

*%% This snippet relies on some internal commands that are not part of the  
 %% most common and stable syntax generally recommended to regular users.*

```

\markup \fontsize #-3 {
  \center-column {
    \vspace #0.3
    Default:
    \vspace #1.7
    \score {

```

```

\new StaffGroup <<
  \new Staff \with {
    \magnifyStaff #1/2
  } { b1 b \bar "|."}
  \new Staff { b b }
>>
}
}
\hspace #2
\center-column {
  "Reverting only the"
  "final bar line:"
  \vspace #1.2
  \score {
    \new StaffGroup <<
      \new Staff \with {
        \magnifyStaff #1/2
      } {
        b1 b
        \revert Staff.BarLine.thick-thickness
        \revert Staff.BarLine.hair-thickness
        \revert Staff.BarLine.kern
        \bar "|."
      }
    \new Staff { b b }
  >>
}
}
\hspace #2
\center-column {
  \line {Cancelling \typewriter "\magnifyStaff"}
  "only for bar lines:"
  \vspace #1.1
  \score {
    \new StaffGroup <<
      \new Staff \with {
        \magnifyStaff #1/2
        #(revert-props 'magnifyStaff 0
          '((BarLine thick-thickness)
            (BarLine hair-thickness)
            (BarLine kern)))
      } { b1 b \bar "|."}
    \new Staff { b b }
  >>
}
}
\hspace #2
\center-column {
  \line {Mimicking \typewriter "\magnifyStaff"}
  "on the other staves:"
  \vspace #1.3
  \score {

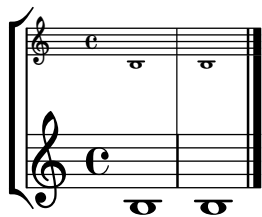
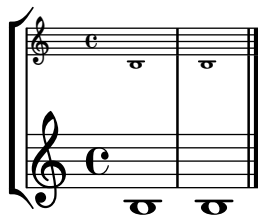
```

```

\new StaffGroup <<
  \new Staff \with {
    \magnifyStaff #1/2
  } { b1 b \bar "|" }
  \new Staff \with {
    #(scale-props 'magnifyStaff 1/2 #t
      '((BarLine thick-thickness)
        (BarLine hair-thickness)
        (BarLine kern)))
  } { b b }
>>
}
}
\hspace #2
\center-column {
  "Applying an intermediate"
  "value to all staves:"
  \vspace #1.3
  \score {
    \new StaffGroup <<
      \new Staff \with {
        \magnifyStaff #1/2
        #(scale-props 'magnifyStaff 3/2 #t
          '((BarLine thick-thickness)
            (BarLine hair-thickness)
            (BarLine kern)))
      } { b1 b \bar "|" }
      \new Staff \with {
        #(scale-props 'magnifyStaff 3/4 #t
          '((BarLine thick-thickness)
            (BarLine hair-thickness)
            (BarLine kern)))
      } { b b }
    >>
  }
}
\layout { indent = 0 }

```

Default:

Reverting only the  
final bar line:Cancelling \magnifyStaff  
only for bar lines:Mimicking \magnifyStaff  
on the other staves:Applying an interme  
value to all staves:

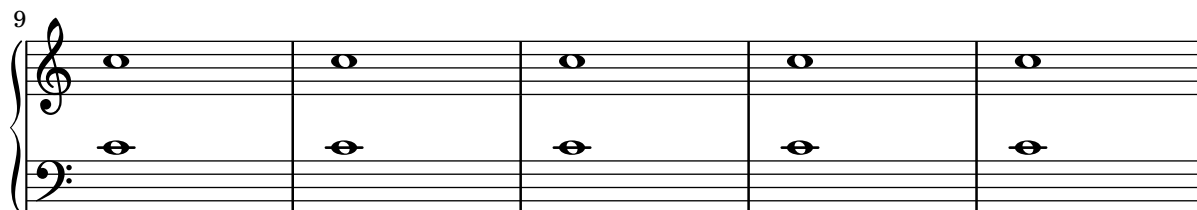
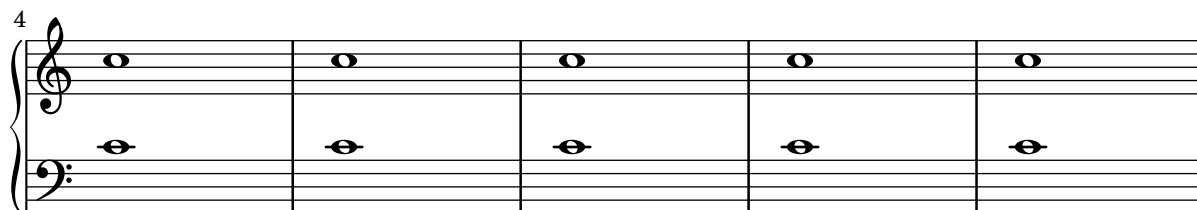
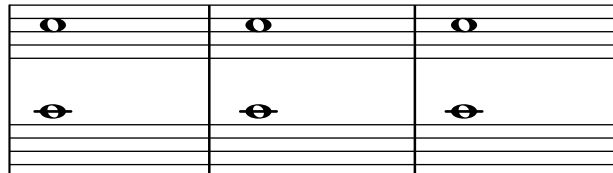
## Removing brace on first line of piano score

This snippet removes the first brace from a PianoStaff or a GrandStaff, together with the clefs. It may be useful when cutting and pasting the engraved image into existing music.

The code uses `\alterBroken` to hide the brace delimiter at the beginning.

```
someMusic = {
  \once \omit Staff.Clef
  \once \omit Staff.TimeSignature
  \repeat unfold 3 c1 \break
  \repeat unfold 5 c1 \break
  \repeat unfold 5 c1
}

\score {
  \new PianoStaff
  <<
    \new Staff = "right" \relative c' { \someMusic
    \new Staff = "left" \relative c' { \clef F \someMusic }
  >>
  \layout {
    indent=75\mm
    \context {
      \PianoStaff
      \alterBroken transparent #'(#t) SystemStartBrace
    }
  }
}
```



## Removing connecting bar lines on StaffGroup, PianoStaff, or GrandStaff

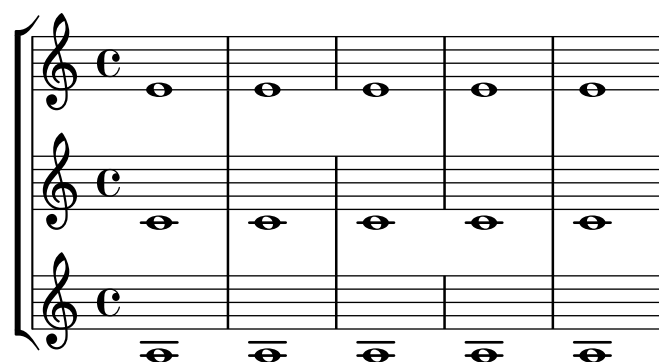
By default, bar lines in `StaffGroup`, `PianoStaff`, or `GrandStaff` contexts are connected between the staves, i.e., a span bar is printed. This behaviour can be overridden on a staff-by-staff basis.

```
\relative c' {
  \new StaffGroup <<
    \new Staff {
      e1 | e
```

```

    \once \override Staff.BarLine.allow-span-bar = ##f
    e1 | e | e
  }
  \new Staff {
    c1 | c | c
    \once \override Staff.BarLine.allow-span-bar = ##f
    c1 | c
  }
  \new Staff {
    a1 | a | a | a | a
  }
}
>>
}

```



## Removing the first empty line

To remove the first empty staff from a score, set the `remove-first` property of the `VerticalAxisGroup` grob to `#t`. 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:
    % \override VerticalAxisGroup.remove-first = ##t
  }
}
\new StaffGroup <<
  \new Staff \relative c' {
    e4 f g a \break
    c1
  }
  \new Staff {
    % To use the setting globally, comment this line,
    % uncomment the line in the \layout block above
    \override Staff.VerticalAxisGroup.remove-first = ##t
    R1 \break
    R
  }
}

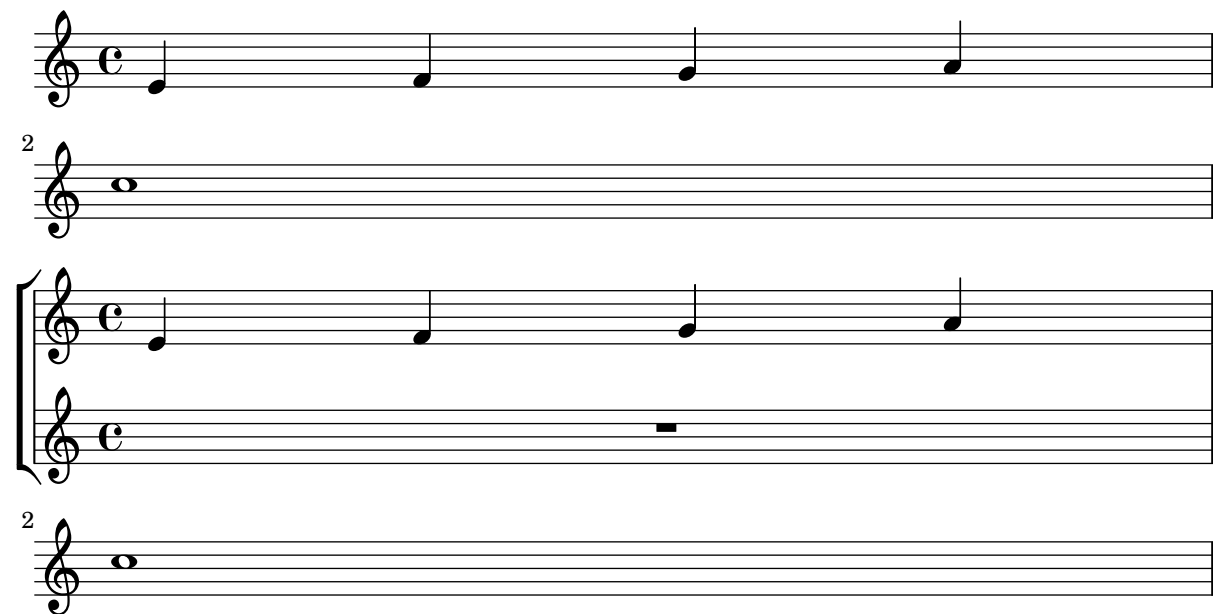
```

&gt;&gt;

```

\new StaffGroup <<
  \new Staff \relative c' {
    e4 f g a \break
    c1
  }
  \new Staff {
    R1 \break
    R
  }
>>

```



## Setting system separators

System separators can be inserted between systems. Any markup can be used, but `\slashSeparator` has been provided as a sensible default.

```

\set-default-paper-size "a5"

```

```

\paper {
  system-separator-markup = \slashSeparator
  tagline = ##f
}

```

```

notes = \relative c' {
  c1 | c \break
  c1 | c \break
  c1 | c
}

```

```

\book {
  \score {
    \new GrandStaff <<
      \new Staff \notes

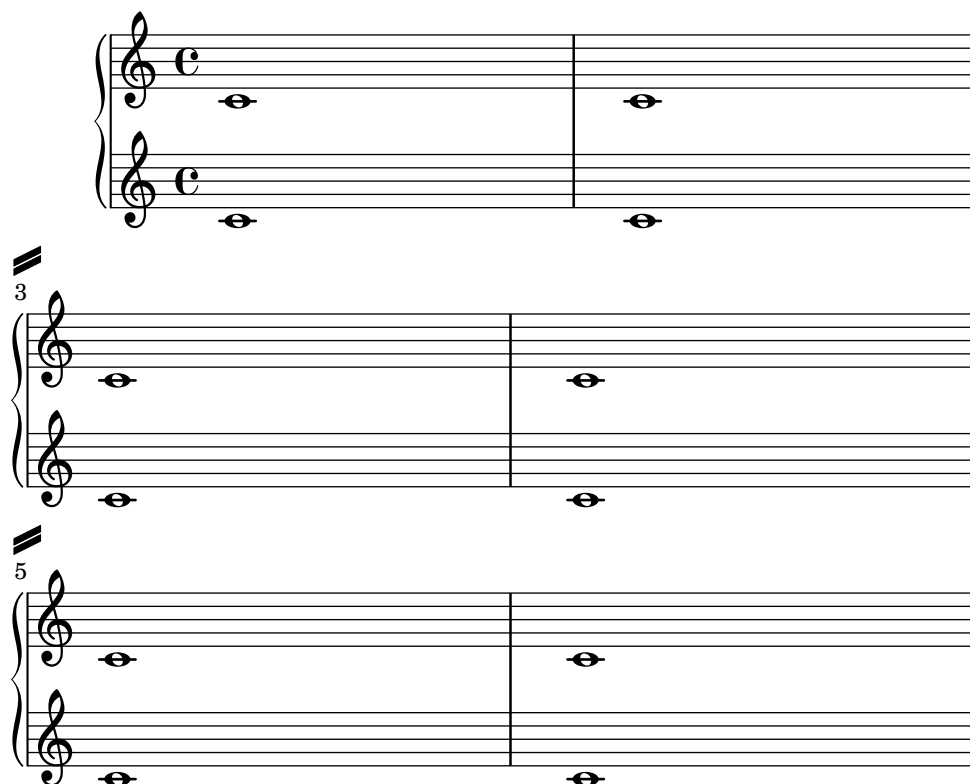
```



```

\new Staff \notes
>>
}
}

```



## Shape individual ties in chords

To shape individual ties in chords use the method demonstrated below.

```

{
  \textMark "Chords can be tied note by note."
  <c'~ e'~ g'~ c''~>2 q
}

{
  \textMark \markup \override #'(baseline-skip . 3) \wordwrap {
    Modifying those ties with \typewriter "\\shape" does not succeed,
    because \typewriter TieColumn positions them on its own behalf,
    ignoring \typewriter "\\shape" input more or less. You may
    circumvent this by setting \typewriter positioning-done to
    \typewriter "#t" -- alas, \typewriter positioning-done is an
    internal property, and setting it to \typewriter "#t" means: all
    positioning is done, don't do anything further. The next example
    demonstrates a case where the positioning is not finished: all tie
    directions are down, and the thickness is not accurate.
  }
  <c'~ e'~ g'~ c''~>2
  \once \override TieColumn.positioning-done = ##t
  q
}

```

```

{
  \textMark "To fix that, enter ties with explicit direction modifiers."
  <c'_~ e'_~ g'_~ c''^~>2
  \once \override TieColumn.positioning-done = ##t
  q
}

{
  \textMark \markup {
    Now you can use \typewriter "\\shape" for each tie as usual. }
  <c'-\shape #'((0 . 0) (0 . -10) (0 . -10) (0 . 0)) _~
    e'-\shape #'((0 . 0) (0 . -5) (0 . -5) (0 . 0)) _~
    g'-\shape #'((0 . 0) (0 . -2) (0 . -2) (0 . 0)) _~
    c''-\shape #'((0 . 0) (0 . 5) (0 . 5) (0 . 0)) ^~
  >2
  \once \override TieColumn.positioning-done = ##t
  q
}

{
  \textMark "This also works at line breaks."
  <c'-\shape #'(((0 . 0) (0 . -10) (0 . -10) (0 . 0))
    ((0 . 0) (0 . -10) (0 . -10) (0 . 0))) _~
    e'-\shape #'(((0 . 0) (0 . -5) (0 . -5) (0 . 0))
    ((0 . 0) (0 . -5) (0 . -5) (0 . 0))) _~
    g'-\shape #'(((0 . 0) (0 . -2) (0 . -2) (0 . 0))
    ((0 . 0) (0 . -2) (0 . -2) (0 . 0))) _~
    c''-\shape #'(((0 . 0) (0 . 5) (0 . 5) (0 . 0))
    ((0 . 0) (0 . 5) (0 . 5) (0 . 0))) ^~
  >2
  \break
  \once \override TieColumn.positioning-done = ##t
  q
}

{
  \textMark \markup {
    It also works with the \typewriter tieWaitForNote property. }
  \set tieWaitForNote = ##t
  c'4-\shape #'((0 . 0) (0 . -10) (0 . -10) (0 . 0)) _~
  e'-\shape #'((0 . 0) (0 . -5) (0 . -5) (0 . 0)) _~
  g'-\shape #'((0 . 0) (0 . -2) (0 . -2) (0 . 0)) _~
  c''-\shape #'((0 . 0) (0 . 5) (0 . 5) (0 . 0)) ^~
  \once \override TieColumn.positioning-done = ##t
  <c' e' g' c''>1
}

\layout {
  indent = 0
  \context {
    \Score

```

```

\override TextMark.padding = #4
\override TextMark.break-align-symbols = #'(left-edge)
}
}

\paper {
  score-system-spacing.padding = 3
}

```

Chords can be tied note by note.



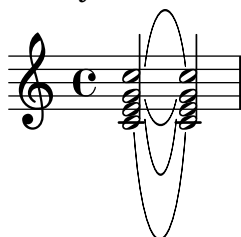
Modifying those ties with `\shape` does not succeed, because `TieColumn` positions them on its own behalf, ignoring `\shape` input more or less. You may circumvent this by setting `positioning-done` to `#t` – alas, `positioning-done` is an internal property, and setting it to `#t` means: all positioning is done, don't do anything further. The next example demonstrates a case where the positioning is not finished: all tie directions are down, and the thickness is not accurate.



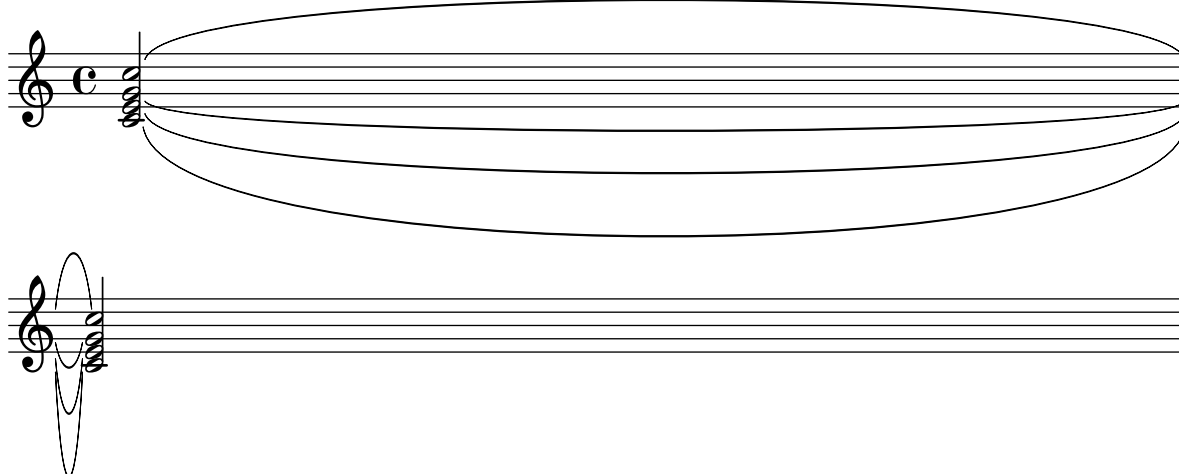
To fix that, enter ties with explicit direction modifiers.



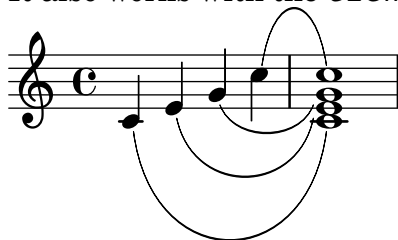
Now you can use `\shape` for each tie as usual.



This also works at line breaks.



It also works with the `tieWaitForNote` property.



## Tick bar lines

‘Tick’ bar lines are often used in music where the bar line is used only for coordination and is not meant to imply any rhythmic stress.

```
\relative c' {
  \set Score.measureBarType = #"'"
  c4 d e f
  g4 f e d
  c4 d e f
  g4 f e d
  \bar "|."
}
```



## Time signature in brackets

The time signature can be enclosed within brackets.

```
\relative c' ' {
  \override Staff.TimeSignature.stencil = #(lambda (grob)
    (bracketify-stencil (ly:time-signature::print grob) Y 0.1 0.2 0.1))
  \time 2/4
  a4 b8 c
}
```



## Time signature in parentheses

The time signature can be enclosed within parentheses.

```
\relative c' ' {
  \override Staff.TimeSignature.stencil = #(lambda (grob)
    (parenthesize-stencil (ly:time-signature::print grob) 0.1 0.4 0.4 0.1))
  \time 2/4
  a4 b8 c
}
```



## Tweaking clef properties

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

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

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

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

  % The following clef changes do not preserve
  % the normal relationship between notes, key signatures
  % and clefs.
  \set Staff.clefGlyph = "clefs.F"
  \set Staff.clefPosition = 2
```

```

c'1
\set Staff.clefGlyph = "clefs.G"
c'1
\set Staff.clefGlyph = "clefs.C"
c'1
\set Staff.clefTransposition = 7
c'1
\set Staff.clefTransposition = 0
\set Staff.clefPosition = 0
c'1

% Return to the normal clef.
\set Staff.middleCPosition = 0
c'1
}

```



## Two \partCombine pairs on one staff

The `\partCombine` function takes two music expressions, each containing a part, and distributes them among four Voice contexts named “one”, “two”, “solo”, and “shared”, depending on when and how the parts are merged into a common voice.

Variants of `\partCombine` are `\partCombineUp` and `\partCombineDown` to produce up-stem and down-stem merging of two voices, respectively. Combining them to squeeze four parts into a single staff, however, need some special setup, which this snippet defines accordingly.

```

customPartCombineUp =
#(define-music-function (part1 part2) (ly:music? ly:music?)
  "Make an up-stem `VoiceBox` context that combines PART1 and PART2.

```

The context is called 'Up'; internally, the function calls `\partCombineUp`.`

```

#{
  \new VoiceBox = "Up" <<
    \context Voice = "one" { \voiceOne }
    \context Voice = "two" { \voiceThree }
    \context Voice = "shared" { \voiceOne }
    \context Voice = "solo" { \voiceOne }
    \context NullVoice = "null" {}
    \partCombine #part1 #part2
  >>
#})

```

```

customPartCombineDown =
#(define-music-function (part3 part4) (ly:music? ly:music?)
  "Make a down-stem `VoiceBox` context that combines PART3 and PART4.

```

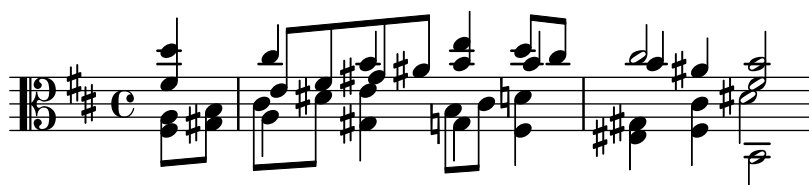
The context is called 'Down'; internally, the function calls `\\partCombineDown`.`

```
#{
  \new VoiceBox = "Down" <<
    \set VoiceBox.soloText = #"Solo III"
    \set VoiceBox.soloIIIText = #"Solo IV"
    \context Voice = "one" { \voiceFour }
    \context Voice = "two" { \voiceTwo }
    \context Voice = "shared" { \voiceFour }
    \context Voice = "solo" { \voiceFour }
    \context NullVoice = "null" {}
    \partCombine #part3 #part4
  >>
#})

soprano = { d'4 | cis' b e' d'8 cis' | cis'2 b }
alto = { fis4 | e8 fis gis ais b4 b | b ais fis2 }
tenor = { a8 b | cis' dis' e'4 b8 cis' d'4 | gis cis' dis'2 }
bass = { fis8 gis | a4 gis g fis | eis fis b,2 }

\new Staff <<
  \key b\minor
  \clef alto
  \partial 4
  \transpose b b' \customPartCombineUp \soprano \alto
  \customPartCombineDown \tenor \bass
>>

\layout {
  \context {
    \Staff
    \accepts "VoiceBox"
  }
  \context {
    \name "VoiceBox"
    \type "Engraver_group"
    \defaultchild "Voice"
    \accepts "Voice"
    \accepts "NullVoice"
  }
}
```



## 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' }
  >> }
}
```



## Using `\autoChange` with more than one voice

Here is a demonstration of how to use `\autoChange` with more than one voice.

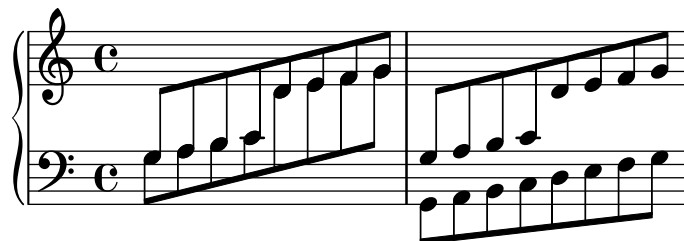
```
\score {
  \new PianoStaff
  <<
    \new Staff = "up" {
      <<
        \set Timing.beamExceptions = #'()
        \set Timing.beatStructure = #'(4)
        \new Voice {
          \voiceOne
          \autoChange
          \relative c' {
            g8 a b c d e f g
            g,.8 a b c d e f g
          }
        }
      >>
    }

    \new Voice {
      \voiceTwo
      \autoChange
      \relative c' {
        g8 a b c d e f g
        g,,.8 a b c d e f g
      }
    }
  >>
}

\new Staff = "down" {
  \clef bass
}
```



```
>>
}
```



## Using mark lines in a Frenched score

Using `MarkLine` contexts (such as in “Placing rehearsal marks other than above the top staff”) in a Frenched score can be problematic if all the staves between two `MarkLines` are removed in one system. The `Keep_alive_together_engraver` can be used within each `StaffGroup` to keep the `MarkLine` alive only as long as the other staves in the group stay alive.

```
bars = {
  \tempo "Allegro" 4=120
  s1*2
  \repeat unfold 5 { \mark \default s1*2 }
  \bar "||"
  \tempo "Adagio" 4=40
  s1*2
  \repeat unfold 8 { \mark \default s1*2 }
  \bar "|."
}

winds = \repeat unfold 120 { c''4 }
trumpet = { \repeat unfold 8 g'2 R1*16 \repeat unfold 4 g'2 R1*8 }
trombone = { \repeat unfold 4 c'1 R1*8 d'1 R1*17 }
strings = \repeat unfold 240 { c''8 }

#(set-global-staff-size 16)
\paper {
  systems-per-page = 5
  ragged-last-bottom = ##f
  tagline = ##f
}

\layout {
  indent = 16\mm
  short-indent = 5\mm
  \context {
    \name MarkLine
    \type Engraver_group
    \consists Output_property_engraver
    \consists Axis_group_engraver
    \consists Mark_engraver
    \consists Metronome_mark_engraver
    \consists Staff_collecting_engraver
    \override VerticalAxisGroup.remove-empty = ##t
    \override VerticalAxisGroup.remove-layer = #'any
```

```

\override VerticalAxisGroup.staff-affinity = #DOWN
\override VerticalAxisGroup.nonstaff-relatedstaff-spacing.padding = 1
keepAliveInterfaces = #'()
}
\context {
  \Staff
  \override VerticalAxisGroup.remove-empty = ##t
  \override VerticalAxisGroup.remove-layer = ##f
}
\context {
  \StaffGroup
  \accepts MarkLine
  \consists Keep_alive_together_engraver
}
\context {
  \Score
  \remove Mark_engraver
  \remove Metronome_mark_engraver
  \remove Staff_collecting_engraver
  \override BarNumber.Y-offset = #3
}
}

\score {
  <<
  \new StaffGroup = "winds" \with {
    instrumentName = "Winds"
    shortInstrumentName = "W."
  } <<
  \new MarkLine \bars
  \new Staff \winds
  >>
  \new StaffGroup = "brass" <<
  \new MarkLine \bars
  \new Staff = "trumpet" \with {
    instrumentName = "Trumpet"
    shortInstrumentName = "Tp."
  } \trumpet
  \new Staff = "trombone" \with {
    instrumentName = "Trombone"
    shortInstrumentName = "Tb."
  } \trombone
  >>
  \new StaffGroup = "strings" \with {
    instrumentName = "Strings"
    shortInstrumentName = "Str."
  } <<
  \new MarkLine \bars
  \new Staff = "strings" { \strings }
  >>
  >>
}

```

**Allegro** (♩ = 120)

**Winds**

**Trumpet**

**Trombone**

**Strings**

**A** **B**

**6**

**W.**

**Str.**

**C** **D**

**11**

**W.**

**Tb.**

**Str.**

**E** **F**

**Adagio** (♩ = 40)

**16**

**W.**

**Str.**

**G** **H**

**21**

**W.**

**Trp.**

**Str.**

**J** **K** **L**

**26**

**W.**

**Str.**

**M** **N**

Detailed description: This musical score is for a symphonic band or orchestra. It features four main parts: Winds, Trumpet, Trombone, and Strings. The score is divided into measures, with measure numbers 6, 11, 16, 21, and 26 marked at the beginning of new systems. The tempo is marked 'Allegro' (♩ = 120) for the first system and 'Adagio' (♩ = 40) for the second system. The score includes various musical notations such as treble and bass clefs, time signatures (C for common time), and dynamic markings. The sections are labeled with letters A through N, indicating different musical themes or sections. The Winds and Strings parts are written in treble clef, while the Trumpet and Trombone parts are written in bass clef. The Strings part is written in a grand staff (treble and bass clef). The score is presented in a clean, professional layout with clear notation and a consistent font.

## Vertically aligned StaffGroups without connecting SystemStartBar

This snippet shows how to achieve vertically aligned StaffGroups with a SystemStartBar for each StaffGroup, but without connecting them.

Note that this only works properly for music that can be printed as a single system.

```

#(set-global-staff-size 15)

\paper {
  ragged-right = ##f
  print-all-headers = ##t
  tagline = ##f
}

\layout {
  indent = 0

  \context {
    \StaffGroup
    \consists Text_mark_engraver
    \consists Staff_collecting_engraver
    systemStartDelimiterHierarchy =
      #'(SystemStartBrace (SystemStartBracket a b))
  }

  \context {
    \Score
    \remove Text_mark_engraver
    \remove Staff_collecting_engraver
    \override SystemStartBrace.style = #'bar-line
    \omit SystemStartBar
    \override SystemStartBrace.padding = #-0.1
    \override SystemStartBrace.thickness = #1.6
    \override StaffGrouper.staffgroup-staff-spacing.basic-distance = #15
  }
}

%%% EXAMPLE

txt =
\lyricmode {
  Wer4 nur den lie -- ben Gott läßt wal2 -- ten4
  und4 hof -- fet auf ihn al -- le Zeit2.
}

% First StaffGroup "exercise"

eI = \relative c' {
  \textMark \markup {
    \bold Teacher:
    This is a simple setting of the choral. Please improve it. }
  \key a \minor
  \time 4/4

```

```

\voiceOne

\partial 4 e4
a b c b
a b gis2
e4\fermata g! g f
e a a gis
a2.\fermata
\bar " : | ."
}

eII = \relative c' {
  \key a \minor
  \time 4/4
  \voiceTwo
  \partial 4 c4
  e e e gis
  a f e2
  b4 b d d
  c c d d
  c2.
  \bar " : | ."
}

eIII = \relative c' {
  \key a \minor
  \time 4/4
  \clef bass
  \voiceOne

  \partial 4 a4
  c b a b
  c d b2
  gis4 g g b
  c a f e
  e2.
}

eIV = \relative c' {
  \key a \minor
  \time 4/4
  \clef bass
  \voiceTwo

  \partial 4 a,4
  a' gis a e
  a, d e2
  e,4\fermata e' b g
  c f d e
  a,2.\fermata
  \bar " : | ."
}

```

```

exercise = \new StaffGroup = "exercise" <<
  \new Staff <<
    \new Voice \eI
    \new Voice \eII
  >>

  \new Lyrics \txt

  \new Staff <<
    \new Voice \eIII
    \new Voice \eIV
  >>
>>

% Second StaffGroup "simple Bach"

sbI = \relative c' {
  \textMark \markup { \bold" Pupil:" Here's my version! }
  \key a \minor
  \time 4/4
  \voiceOne

  \partial 4 e4
  a b c b
  a b gis2
  e4\fermata g! g f
  e a a gis
  a2.\fermata
  \bar ":|."
}

sbII = \relative c' {
  \key a \minor
  \time 4/4
  \voiceTwo
  \partial 4 c8 d
  e4 e e8 f g4
  f f e2
  b4 b8 c d4 d
  e8 d c4 b8 c d4
  c2.
  \bar ":|."
}

sbIII = \relative c' {
  \key a \minor
  \time 4/4
  \clef bass
  \voiceOne

```

```

\partial 4 a8 b
c4 b a b8 c
d4 d8 c b2
gis4 g g8 a b4
b a8 g f4 e
e2.
}

sbIV = \relative c' {
  \key a \minor
  \time 4/4
  \clef bass
  \voiceTwo

  \partial 4 a,4
  a' gis a e
  f8 e d4 e2
  e,4\fermata e' b a8 g
  c4 f8 e d4 e
  a,2.\fermata
  \bar ":|."
}

simpleBach = \new StaffGroup = "simple Bach" <<
  \new Staff <<
    \new Voice \sbI
    \new Voice \sbII
  >>

  \new Lyrics \txt

  \new Staff <<
    \new Voice \sbIII
    \new Voice \sbIV
  >>
>>

% Third StaffGroup "chromatic Bach"

cbI = \relative c' {
  \textMark \markup {
    \bold "Teacher:"
    \column {
      "Well, you simply copied and transposed a version of J.S.Bach."
      "Do you know this one?"
    }
  }
}

\key a \minor
\time 4/4
\voiceOne

```

```

\partial 4 e4
a b c b
a b gis4. fis8
e4\fermata g! g f
e a a8 b gis4
a2.\fermata
\bar " : | ."
}

cbII = \relative c' {
  \key a \minor
  \time 4/4
  \voiceTwo

  \partial 4 c8 d
  e4 e e8 fis gis4
  a8 g! f!4 e2
  b4 e e d
  d8[ cis] d dis e fis e4
  e2.
  \bar " : | ."
}

cbIII = \relative c' {
  \key a \minor
  \time 4/4
  \clef bass
  \voiceOne

  \partial 4 a8 b
  c[ b] a gis8 a4 d,
  e8[ e'] d c b4. a8
  gis4 b c d8 c
  b[ a] a b c b b c16 d
  c2.
}

cbIV = \relative c' {
  \key a \minor
  \time 4/4
  \clef bass
  \voiceTwo

  \partial 4 a4
  c, e a, b
  c d e2
  e4\fermata e a b8 c
  gis[ g] fis f e dis e4
  a,2.\fermata
  \bar " : | ."
}

```



```

chromaticBach = \new StaffGroup = "chromatic Bach" <<
  \new Staff <<
    \new Voice \cbI
    \new Voice \cbII
  >>

  \new Lyrics \txt

  \new Staff <<
    \new Voice \cbIII
    \new Voice \cbIV
  >>
>>

% Score

\score {
  <<
    \exercise
    \simpleBach
    \chromaticBach
  >>

  \header {
    title = \markup \column {
      \combine \null \vspace #1
      "Exercise: Improve the given choral"
      " "
    }
  }

  \layout {
    \context {
      \Lyrics
      \override LyricText.X-offset = #-1
    }
  }
}

```

### Exercise: Improve the given choral

**Teacher:** This is a simple setting of the choral. Please improve it.

Wer nur den lie - ben Gott läßt wal - ten und hof - fet auf ihn al - le Zeit

**Pupil:** Here's my version!

Wer nur den lie - ben Gott läßt wal - ten und hof - fet auf ihn al - le Zeit

**Teacher:** Well, you simply copied and transposed a version of J.S.Bach.  
Do you know this one?

Wer nur den lie - ben Gott läßt wal - ten und hof - fet auf ihn al - le Zeit

## Volta below chords

By adding the `Volta_engraver` to the relevant staff, volte can be put below chords.

```
\score {
  <<
    \chords { c1 c1 }
    \new Staff \with { \consists "Volta_engraver" }
    {
      \repeat volta 2 { c'1 \alternative { c' } }
    }
  >>
  \layout {
    \context {
      \Score
      \remove "Volta_engraver"
    }
  }
}
```

C C 1.2.

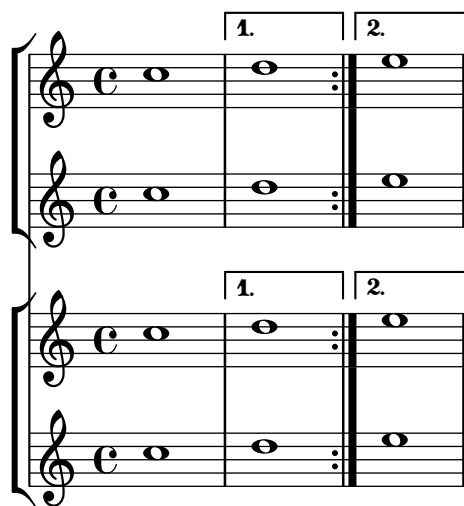
## Volta brackets in multiple staves

By adding the `Volta_engraver` to the relevant staff, volte can be put over staves other than the topmost one in a score.

`\repeat` and related commands should be present in all staves.

```
voltaMusic = \relative c'' {
  \repeat volta 2 {
    c1
    \alternative {
      \volta 1 { d1 }
      \volta 2 { e1 }
    }
  }
}

<<
  \new StaffGroup <<
    \new Staff \voltaMusic
    \new Staff \voltaMusic
  >>
  \new StaffGroup <<
    \new Staff \with { \consists "Volta_engraver" }
      \voltaMusic
    \new Staff \voltaMusic
  >>
>>
```



## 7 Editorial annotations

See also Section “Editorial annotations” in *Notation Reference*.

### Adding fingerings to a score

Fingering instructions can be entered using a simple syntax.

```
\relative c' ' {
  c4-1 d-2 f-4 e-3
}
```



### Adding links to objects

To add a link to a grob stencil you can use `add-link` as defined here. It works both with `\override` and `\tweak`.

Drawback: point-and-click is disturbed for the linked grobs.

Limitation: Works for PDF only.

The linked objects are colored with a separate command.

```
#(define (add-link url-strg)
  (lambda (grob)
    (let* ((stil (ly:grob-property grob 'stencil)))
      (if (ly:stencil? stil)
        (let* ((x-ext (ly:stencil-extent stil X))
              (y-ext (ly:stencil-extent stil Y))
              (url-expr `(url-link ,url-strg ,x-ext ,y-ext))
              (new-stil
                (ly:stencil-add
                 (ly:make-stencil url-expr x-ext y-ext)
                 stil)))
          (ly:grob-set-property! grob 'stencil new-stil))))))
```

```
%%% test
```

```
%% For easier maintenance of this snippet the URL is formatted to use the
%% actually used LilyPond version.
%% Of course a literal URL would work as well.
```

```
#(define major.minor-version
  (string-join (take (string-split (lilypond-version) #\.) 2) "."))

urlI =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/writing-pitches"
  major.minor-version)

urlII =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/rhythms"
```

```

major.minor-version)

urlIII =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/note-heads"
  major.minor-version)

urlIV =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/beams"
  major.minor-version)

urlV =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/note-head-styles"
  major.minor-version)

urlVI =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/writing-pitches"
  major.minor-version)

\relative c' {
  \key cis \minor

  \once \override Staff.Clef.color = #green
  \once \override Staff.Clef.after-line-breaking =
    #(add-link urlI)

  \once \override Staff.TimeSignature.color = #green
  \once \override Staff.TimeSignature.after-line-breaking =
    #(add-link urlII)

  \once \override NoteHead.color = #green
  \once \override NoteHead.after-line-breaking =
    #(add-link urlIII)

  cis'1
  \once \override Beam.color = #green
  \once \override Beam.after-line-breaking =
    #(add-link urlIV)
  cis8 dis e fis gis2
  <gis,
    \tweak Accidental.color #green
    \tweak Accidental.after-line-breaking #(add-link urlVI)
    \tweak color #green
    \tweak after-line-breaking #(add-link urlV)
    \tweak style #'harmonic
  bis
  dis
  fis
  >1

```

```
<cis, cis' e>
}
```



## Adding markups in a tablature

By default, markups are not displayed in a tablature.

To make them appear, revert the stencil property of the TextScript grob in the TabStaff context.

```
high = { r4 r8 <g c'> q r8 r4 }
low = { c4 r4 c8 r8 g,8 b, }
pulse = { s8^"1" s8^"&" s8^"2" s8^"&" s8^"3" s8^"&" s8^"4" s8^"&" }
```

```
\score {
  \new TabStaff {
    \repeat unfold 2 << \high \ \ \low \ \ \pulse >>
  }
  \layout {
    \context {
      \TabStaff
      \clef moderntab
      \revert TextScript.stencil
      \override TextScript.font-series = #'bold
      \override TextScript.font-size = #-2
      \override TextScript.color = #red
    }
    \context {
      \Score
      \proportionalNotationDuration = #1/8
    }
  }
}
```

	1	&	2	&	3	&	4	&	1	&	2	&	3	&	4	&
<b>T</b>																
<b>A</b>																
<b>B</b>	3				3				2	3				3		2

## Allowing fingerings to be printed inside the staff

By default, vertically oriented fingerings are positioned outside the staff; that behavior, however, may be disabled. Attention needs to be paid to situations where fingerings and stems are in the same direction: by default, fingerings will avoid only beamed stems. That setting can be changed to avoid no stems or all stems; the following example demonstrates these two options, as well as how to go back to the default behavior.

```
\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
  a8[-1 b]-2 g-0 r
```

```

\override Fingering.add-stem-support = ##f
a[-1 b]-2 g-0 r
\override Fingering.add-stem-support = ##t
a[-1 b]-2 g-0 r
\override Fingering.add-stem-support = #only-if-beamed
a[-1 b]-2 g-0 r
}

```



## Alternative bar numbering

Setting the `alternativeNumberingStyle` context property, two additional methods are available for enumerating bar numbers in repeats.

```

music = \relative c' {
  \repeat volta 3 {
    c4 d e f |
    \alternative {
      \volta 1 { c4 d e f | c2 d \break }
      \volta 2 { f4 g a b | f4 g a b | f2 a | \break }
      \volta 3 { c4 d e f | c2 d } } }
  c1 \bar " | ."
}

```

```

\markup "default"
{
  \music
}

```

```

\markup \typewriter "numbers"
{
  \set Score.alternativeNumberingStyle = #'numbers
  \music
}

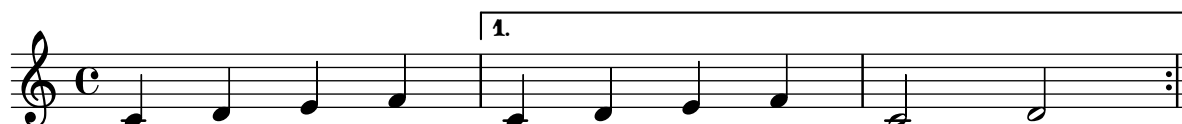
```

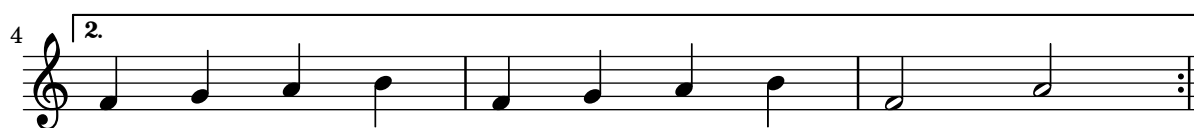
```

\markup \typewriter "numbers-with-letters"
{
  \set Score.alternativeNumberingStyle = #'numbers-with-letters
  \music
}

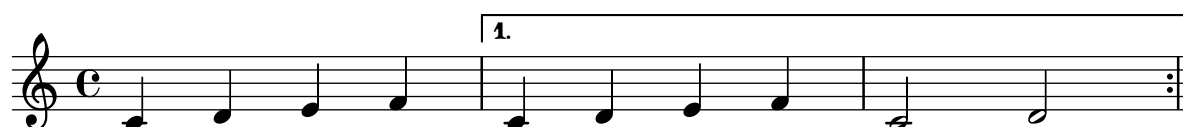
```

default

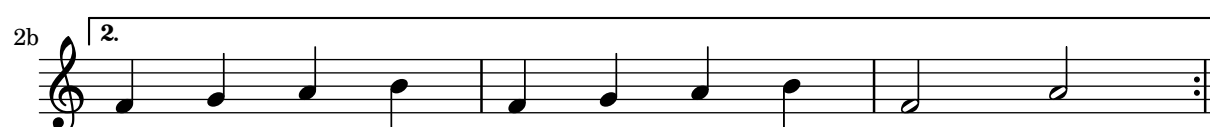




numbers



numbers-with-letters



## Analysis brackets above the staff

Simple horizontal analysis brackets are added below the staff by default. The following example shows a way to place them above the staff instead.

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

\relative c'' {
  \once \override HorizontalBracket.direction = #UP
  c2\startGroup
  d2\stopGroup
}
```



}



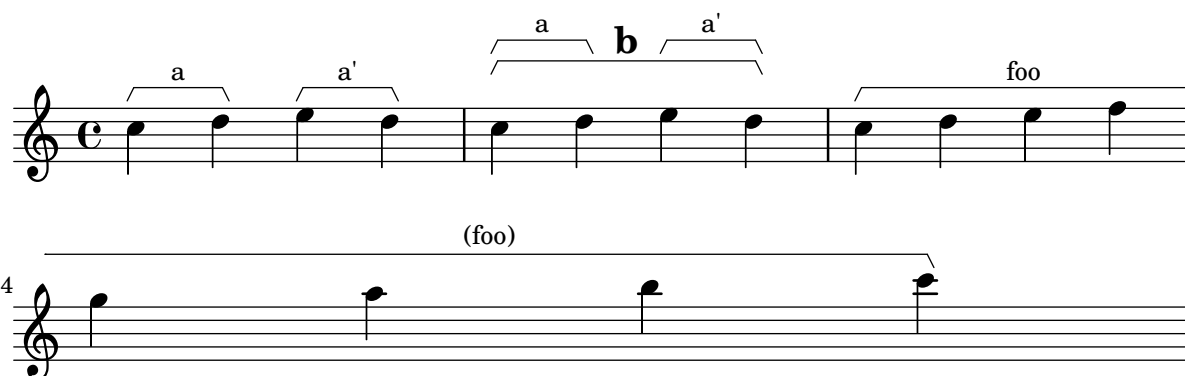
## Analysis brackets with labels

Text markup may be added to analysis brackets using the `text` property of the `HorizontalBracketText` grob. Adding different texts to brackets beginning at the same time requires the `\tweak` command.

Bracket text gets parenthesized after a line break.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
    \override HorizontalBracket.direction = #UP
  }
}

{
  \once\override HorizontalBracketText.text = "a"
  c''\startGroup d''\stopGroup
  \once\override HorizontalBracketText.text = "a'"
  e''\startGroup d''\stopGroup |
  c''-\tweak HorizontalBracketText.text
    \markup \bold \huge "b" \startGroup
    -\tweak HorizontalBracketText.text "a" \startGroup
    d''\stopGroup
    e''-\tweak HorizontalBracketText.text "a'" \startGroup
    d''\stopGroup\stopGroup |
  c''-\tweak HorizontalBracketText.text foo \startGroup
    d'' e'' f'' | \break
    g'' a'' b'' c'''\stopGroup
}
```



## Applying note head styles depending on the step of the scale

The `shapeNoteStyles` property can be used to define various note head styles for each step of the scale (as set by the key signature or the tonic property).

This property requires a set of symbols, which can be purely arbitrary (geometrical expressions such as `triangle`, `cross`, and `xcircle` are allowed) or based on old American engraving tradition (some latin note names are also allowed).

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

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

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

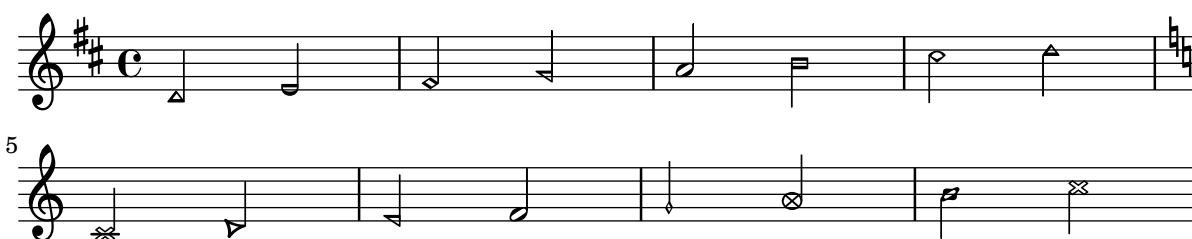
\new Staff {
  \transpose c d
  \relative c' {
    \set shapeNoteStyles = ##(do re mi fa
                          #f la ti)

    \fragment
  }

  \break

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

    \fragment
  }
}
```



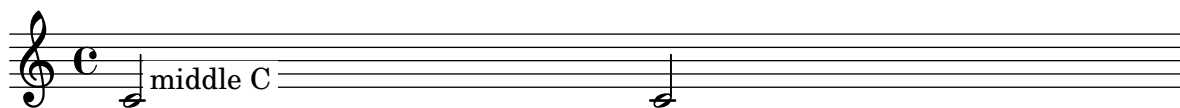
## Blanking staff lines using the `\whiteout` command

The `\whiteout` command underlays a markup with a white box. Since staff lines are in a lower layer than most other grobs, this white box will not overlap any other grob.

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

\relative c' {
  \override TextScript.extra-offset = #'(2 . 4)
  c2-\markup { \whiteout \pad-markup #0.5 "middle C" } c
```

}

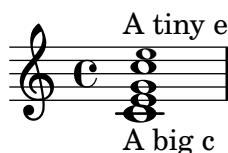


## Changing a single note's size in a chord

Individual note heads in a chord can be modified with the `\tweak` command inside a chord, by altering the `font-size` property.

Inside the chord (within the brackets `< >`), before the note to be altered, place the `\tweak` command, followed by `font-size` and define the proper size like `#-2` (a tiny note head).

```
\relative c' {
  <\tweak font-size #+2 c e g c
    \tweak font-size #-2 e>1
  ~\markup { A tiny e }_~\markup { A big c }
}
```



## Changing the appearance of a slur from solid to dotted or dashed

The appearance of slurs may be changed from solid to dotted or dashed.

```
\relative c' {
  c4( d e c)
  \slurDotted
  c4( d e c)
  \slurSolid
  c4( d e c)
  \slurDashed
  c4( d e c)
  \slurSolid
  c4( d e c)
}
```



## Coloring notes depending on their pitch

It is possible to color note heads depending on their pitch and/or their names: the function used in this example even makes it possible to distinguish enharmonics.

*% Association list of pitches to colors.*

```
#(define color-mapping
  (list
    (cons (ly:make-pitch 0 0 NATURAL) (x11-color 'red))
```

```

(cons (ly:make-pitch 0 0 SHARP) (x11-color 'green))
(cons (ly:make-pitch 0 1 FLAT) (x11-color 'green))
(cons (ly:make-pitch 0 2 NATURAL) (x11-color 'red))
(cons (ly:make-pitch 0 2 SHARP) (x11-color 'green))
(cons (ly:make-pitch 0 3 FLAT) (x11-color 'red))
(cons (ly:make-pitch 0 3 NATURAL) (x11-color 'green))
(cons (ly:make-pitch 0 4 SHARP) (x11-color 'red))
(cons (ly:make-pitch 0 5 NATURAL) (x11-color 'green))
(cons (ly:make-pitch 0 5 FLAT) (x11-color 'red))
(cons (ly:make-pitch 0 6 SHARP) (x11-color 'red))
(cons (ly:make-pitch 0 1 NATURAL) (x11-color 'blue))
(cons (ly:make-pitch 0 3 SHARP) (x11-color 'blue))
(cons (ly:make-pitch 0 4 FLAT) (x11-color 'blue))
(cons (ly:make-pitch 0 5 SHARP) (x11-color 'blue))
(cons (ly:make-pitch 0 6 FLAT) (x11-color 'blue)))

% Compare pitch and alteration (not octave).
#(define (pitch-equals? p1 p2)
  (and
    (= (ly:pitch-alteration p1) (ly:pitch-alteration p2))
    (= (ly:pitch-notename p1) (ly:pitch-notename p2))))

#(define (pitch-to-color pitch)
  (let ((color (assoc pitch color-mapping pitch-equals?)))
    (if color
      (cdr color))))

#(define (color-notehead grob)
  (pitch-to-color
    (ly:event-property (event-cause grob) 'pitch)))

\score {
  \new Staff \relative c' {
    \override NoteHead.color = #color-notehead
    c8 b d dis ees f g aes
  }
}

```



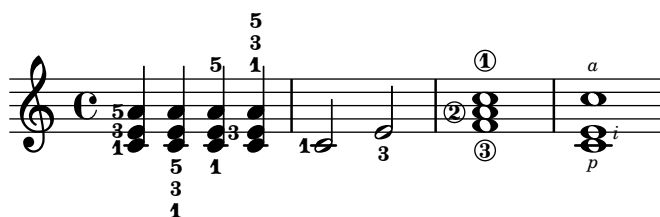
## Controlling the placement of chord fingerings

The placement of fingering numbers can be controlled precisely by using the property `fingeringOrientation`. For fingering orientation to apply, the fingering command must be used within a chord construct (`<...>`), even for single notes. Orientation for string numbers and right-hand fingerings may be controlled in a similar way by using the properties `stringNumberOrientation` and `strokeFingerOrientation`, respectively.

These properties can be set to a list of one to three values. They control whether fingerings may be placed above (if `up` appears in the list), below (if `down` appears), to the left (if `left` appears), or to the right (if `right` appears). Conversely, if a location is not listed, no fingering

is placed there. LilyPond takes these constraints and works out the best placement for the fingering of the notes of the following chords. Note that `left` and `right` are mutually exclusive – fingerings may be placed only on one side or the other, not both.

```
\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
  \set stringNumberOrientations = #'(up left down)
  <f\3 a\2 c\1>1
  \set strokeFingerOrientations = #'(down right up)
  <c\rightHandFinger 1 e\rightHandFinger 2 c'\rightHandFinger 4 >
}
```



## 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 first argument of `\after` (which is a duration) adjusts the horizontal position.

```
\relative c' {
  \after 2*2/3 \turn c2( d4) r |
  \after 4 \turn c4.( d8)
  \after 4
  {
    \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
    \once \hideNotes
    cis8\turn \noBeam
  }
  d4.( e8)
}
```



## Creating blank staves

To create blank staves, generate empty measures then remove the `Bar_number_engraver` from the `Score` context, and the `Time_signature_engraver`, `Clef_engraver` and `Bar_engraver` from the `Staff` context.

```
#(set-global-staff-size 10) % for the documentation
% #(set-global-staff-size 20) % for letter and A4
```

```
\book {
  \score {
    { \repeat unfold 12 { s1 \break } }

    \layout {
      indent = 0
      \context {
        \Staff
        \remove "Time_signature_engraver"
        \remove "Clef_engraver"
        \remove "Bar_engraver"
      }
      \context {
        \Score
        \remove "Bar_number_engraver"
      }
    }
  }
}
```

```
% for the documentation
```

```
\paper {
  #(set-paper-size "a6")
  ragged-last-bottom = ##f
  line-width = 90\mm
  left-margin = 7.5\mm
  bottom-margin = 5\mm
  top-margin = 5\mm
  tagline = ##f
}
```

```
% uncomment these lines for "letter" size
```

```
%{
```

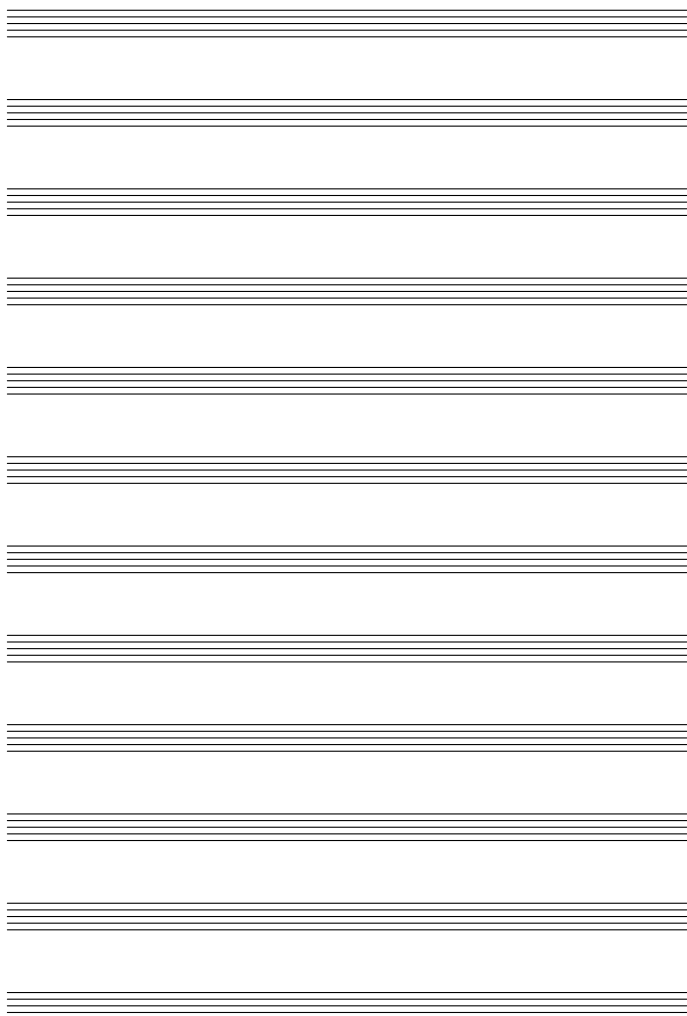
```
\paper {
  #(set-paper-size "letter")
  ragged-last-bottom = ##f
  line-width = 7.5\in
  left-margin = 0.5\in
  bottom-margin = 0.25\in
  top-margin = 0.25\in
}
```

```

    tagline = ##f
  }
%}

% uncomment these lines for "A4" size
%{
\paper {
  #(set-paper-size "a4")
  ragged-last-bottom = ##f
  line-width = 180\mm
  left-margin = 15\mm
  bottom-margin = 10\mm
  top-margin = 10\mm
  tagline = ##f
}
%}
}

```



## Creating double-digit fingerings

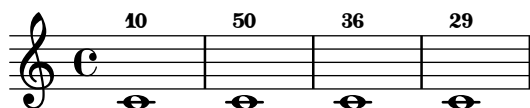
Creating fingerings larger than 5 is possible.

```
\relative c' {
```

```

c1-10
c1-50
c1-36
c1-29
}

```



## Default direction of stems on the center line of the staff

The default direction of stems on the center line of the staff is set by the Stem property `neutral-direction`.

```

\relative c' {
  a4 b c b
  \override Stem.neutral-direction = #up
  a4 b c b
  \override Stem.neutral-direction = #down
  a4 b c b
}

```



## Different font size settings for `instrumentName` and `shortInstrumentName`

Choose different font sizes for `instrumentName` and `shortInstrumentName` as a context override.

```

InstrumentNameFontSize =
#(define-music-function (font-size-pair) (pair?)
  "Set the font size of `InstrumentName` grobs.

```

The first value of `FONT-SIZE-PAIR` sets the font size of the initial ``instrumentName`` property, the second value sets the font size of ``shortInstrumentName``."

```

;; This code could be changed or extended to set different values
;; for each occurrence of `shortInstrumentName`.
#{
  \override InstrumentName.after-line-breaking =
    #(lambda (grob)
      (let* ((orig (ly:grob-original grob))
             (siblings (if (ly:grob? orig)
                           (ly:spanner-broken-into orig)
                           '()))
             (when (pair? siblings)
               (ly:grob-set-property! (car siblings)
                                       'font-size (car font-size-pair))
               (for-each
                (lambda (g)
                  (ly:grob-set-property! g
                                          'font-size (cdr font-size-pair)))
                siblings)))
    )

```



```

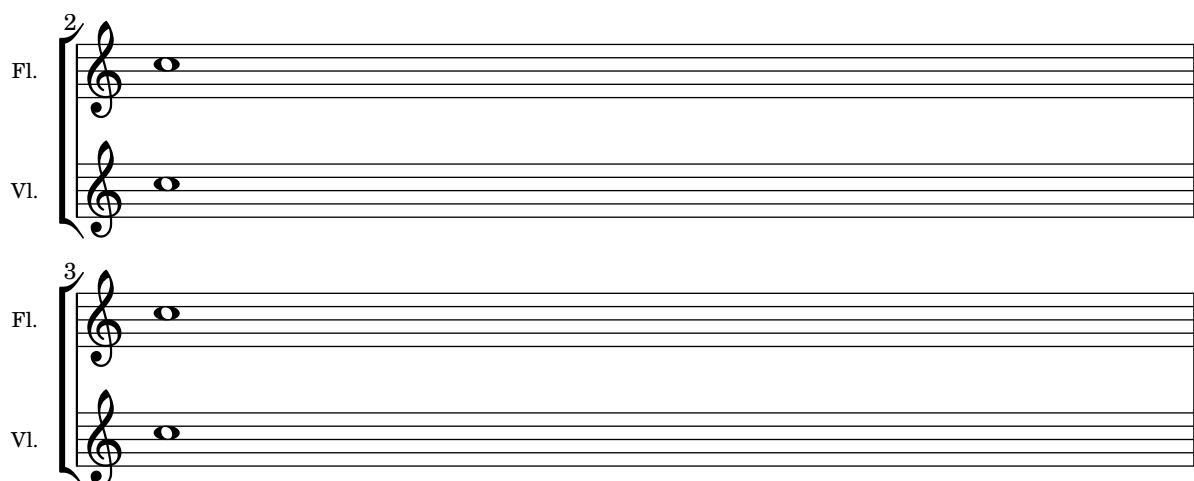
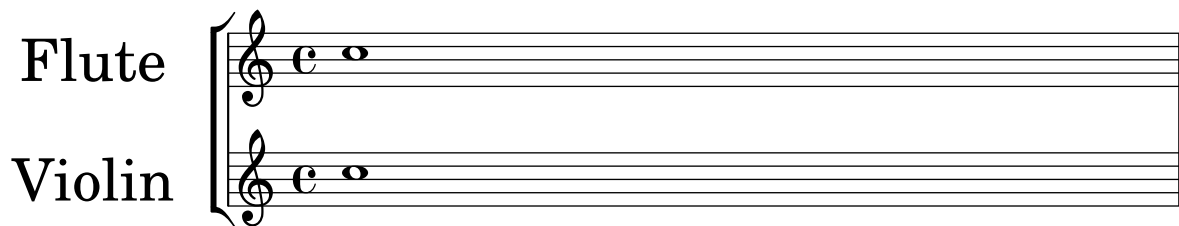
                                (cdr siblings))))))
#})

\layout {
  indent = 3\cm
  short-indent = 0.8\cm

  \context {
    \Staff
    \InstrumentNameFontSize #'(6 . -3)
  }
}

\new StaffGroup <<
  \new Staff \with {
    instrumentName = "Flute"
    shortInstrumentName = "Fl." } {
    c''1 \break c'' \break c'' }
  \new Staff \with {
    instrumentName = "Violin"
    shortInstrumentName = "Vl." } {
    c''1 \break c'' \break c'' }
>>

```



## Drawing boxes around grobs

The `stencil` property can be overridden to draw a box around arbitrary grobs, either using `\override` or `\tweak`.

```

\relative c'' {
  \once \override TextScript.stencil =
    #(make-stencil-boxer 0.1 0.3 ly:text-interface::print)

```

```

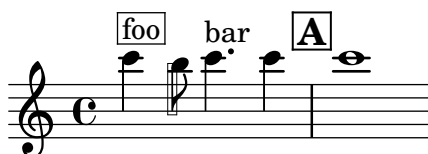
c'4~"foo"

\tweak Stem.stencil
  #(make-stencil-boxer 0.05 0.25 ly:stem::print)
b8

c4.~"bar" c4

\override Score.RehearsalMark.stencil =
  #(make-stencil-boxer 0.15 0.3 ly:text-interface::print)
\mark \default
c1
}

```



## Drawing circles around note heads

A circle can be drawn around a note head by providing a custom Scheme function to temporarily override the stencil property.

```

circle = \tweak NoteHead.stencil
  #(lambda (grob)
    (let* ((note (ly:note-head::print grob))
      (combo-stencil (ly:stencil-add
        note
        (circle-stencil note 0.1 0.8))))
      (ly:make-stencil (ly:stencil-expr combo-stencil)
        (ly:stencil-extent note X)
        (ly:stencil-extent note Y))))
  \etc

{ \circle c' ' }

```



## Drawing circles around various objects

The `\circle` command draws circles around `\markup` objects. For other objects, specific tweaks may be required, as demonstrated for rehearsal marks and measure numbers.

```

\relative c' {
  c1
  \set Score.rehearsalMarkFormatter =
    #(lambda (mark context)
      (make-circle-markup (format-mark-numbers mark context)))
  \mark \default

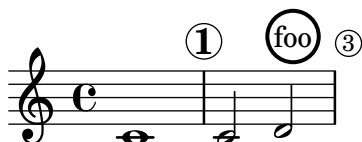
  c2 d~\markup {
    \override #'(thickness . 3) {

```

```

    \circle foo
  }
}
\override Score.BarNumber.break-visibility = #all-visible
\override Score.BarNumber.stencil =
  #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
}

```



## Embedding native PostScript in a `\markup` block

PostScript code can be directly inserted inside a `\markup` block.

In general it is recommended to use LilyPond's native graphical markup commands like `\polygon` instead, which can be used with all LilyPond backends.

```

\relative c' ' {
  a2-\markup \postscript "0 3 moveto
                        5 2 rlineto
                        stroke"
  -\markup \postscript "[1 1] 0 setdash
                        0 0 moveto
                        5 2 rlineto
                        stroke"
  b2-\markup \postscript "1 1 moveto
                        0 0 1 2 8 4 10 2 rcurveto
                        stroke"
  a'1
}

```



## Generate special note head shapes

When a note head with a special shape cannot easily be generated with graphic markup, a drawing specification for `ly:make-stencil` can be used to generate the shape. This example gives an example for a parallelogram-shaped note head.

Unfortunately, the available commands in a drawing specification are currently not documented (<https://gitlab.com/lilypond/lilypond/-/issues/6874>); in any case, the used path sub-command has the following signature, quite similar to the `make-path-stencil` Scheme function.

```
(path thickness command-list line-cap-style line-join-style fill)
```

The commands in *command-list* resemble PostScript drawing commands but with arguments after the command name.

```

parallelogram =
  #(ly:make-stencil
    '(path 0.1
      (rmoveto 0 0.25

```

```

        lineto 1.2 0.75
        lineto 1.2 -0.25
        lineto 0 -0.75
        lineto 0 0.25)
    round
    round
    #t)
    (cons -0.05 1.25)
    (cons -.75 .75))

myNoteHeads = \override NoteHead.stencil = \parallelogram
normalNoteHeads = \revert NoteHead.stencil

\relative c'' {
  \myNoteHeads
  g4 d'
  \normalNoteHeads
  <f, \tweak stencil \parallelogram b e>4 d
}

```



## Grid lines: changing their appearance

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

```

\new ChoirStaff <<
  \new Staff {
    \relative c'' {
      \stemUp
      c'4. d8 e8 f g4
    }
  }
  \new Staff {
    \relative c {
      % this moves them up one staff space from the default position
      \override Score.GridLine.extra-offset = #'(0.0 . 1.0)
      \stemDown
      \clef bass
      \once \override Score.GridLine.thickness = 5.0
      c4
      \once \override Score.GridLine.thickness = 1.0
      g'4
      \once \override Score.GridLine.thickness = 3.0
      f4
      \once \override Score.GridLine.thickness = 5.0
      e4
    }
  }
}
>>

```

```

\layout {
  \context {
    \Staff
    % set up grids
    \consists "Grid_point_engraver"
    % set the grid interval to one quarter note
    gridInterval = #1/4
  }
  \context {
    \Score
    \consists "Grid_line_span_engraver"
    % this moves them to the right half a staff space
    \override NoteColumn.X-offset = -0.5
  }
}

```



## Grid lines: emphasizing rhythms and notes synchronization

Regular vertical lines can be drawn between staves to show note synchronization; however, in case of monophonic music, you may want to make the second staff invisible, and make the lines shorter like in this snippet.

```

\new ChoirStaff {
  \relative c'' <<
  \new Staff {
    \time 12/8
    \stemUp
    c4. d8 e8 f g4 f8 e8. d16 c8
  }
  \new Staff {
    % hides staff and notes so that only the grid lines are visible
    \hideNotes
    \hide Staff.BarLine
    \override Staff.StaffSymbol.line-count = #0
    \hide Staff.TimeSignature
    \hide Staff.Clef

    % dummy notes to force regular note spacing
    \once \override Score.GridLine.thickness = #4.0
    c8 c c
    \once \override Score.GridLine.thickness = #3.0
    c8 c c
    \once \override Score.GridLine.thickness = #4.0
    c8 c c
  }
}

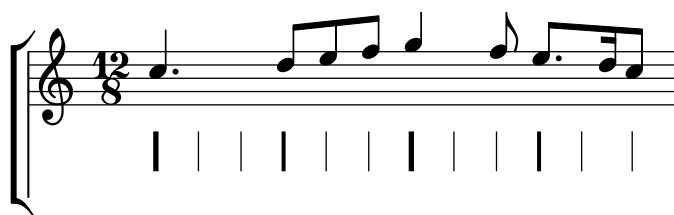
```

```

\once \override Score.GridLine.thickness = #3.0
c8 c c
}
>>
}

\layout {
  \context {
    \Score
    \consists "Grid_line_span_engraver"
    % center grid lines horizontally below note heads
    \override NoteColumn.X-offset = #-0.5
  }
  \context {
    \Staff
    \consists "Grid_point_engraver"
    gridInterval = #1/8
    % set line length and positioning:
    % two staff spaces above center line on hidden staff
    % to four spaces below center line on visible staff
    \override GridPoint.Y-extent = #'(2 . -4)
  }
}

```



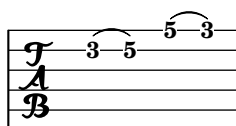
## Hammer-on and pull-off

Hammer-on and pull-off can be obtained using slurs.

```

\new TabStaff {
  \relative c' {
    d4( e\2)
    a( g)
  }
}

```



## Hammer-on and pull-off using chords

When using hammer-on or pull-off with chorded notes, only a single arc is drawn. However “double arcs” are possible by setting the `doubleSlurs` property to `#t`.

```

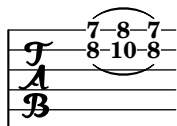
\new TabStaff {
  \relative c' {

```

```

% chord hammer-on and pull-off
\set doubleSlurs = ##t
<g' b>8( <a c> <g b>)
}
}

```



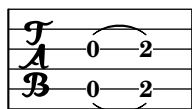
## Hammer-on and pull-off using voices

The arc of hammer-on and pull-off is upwards in voices one and three and downwards in voices two and four:

```

\new TabStaff {
  \relative c' {
    << { \voiceOne g2( a) }
    \\ { \voiceTwo a,( b) }
    >> \oneVoice
  }
}

```



## 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'
}

```



## Marking notes of spoken parts with a cross on the stem (Sprechstimme)

This example shows how to put crosses on stems. Mark the beginning of a spoken section with the command `\speakOn` and end it with `\speakOff`.

```

\speakOn = \override Stem.stencil =
  #(lambda (grob)
    (let* ((x-parent (ly:grob-parent grob X))
          (is-rest? (ly:grob? (ly:grob-object x-parent 'rest))))
      (if is-rest?
          empty-stencil

```

```

      (ly:stencil-combine-at-edge
        (ly:stem::print grob)
        Y
        (- (ly:grob-property grob 'direction))
        (grob-interpret-markup
  grob
          (markup #:center-align #:fontsize -4
            #:musicglyph "noteheads.s2cross")))
      -1.7))))

speakOff = \revert Stem.stencil

\new Staff {
  \relative c'' {
    a4 b a c
    \speakOn
    g4 f r g8 a
    b4 r r8 d e4
    \speakOff
    c4 a g f
  }
}

```



## Measure counters

This snippet demonstrates the use of the `Measure_counter_engraver` to number groups of successive measures. Any stretch of measures may be numbered, whether consisting of repetitions or not.

The engraver must be added to the appropriate context. Here, a `Staff` context is used; another possibility is a `Dynamics` context.

The counter is begun with `\startMeasureCount` and ended with `\stopMeasureCount`. Numbering will start by default with 1, but this behavior may be modified by overriding the `count-from` property.

When a measure extends across a line break, the number will appear twice, the second time in parentheses.

```

\layout {
  \context {
    \Staff
    \consists #Measure_counter_engraver
  }
}

\new Staff {
  \startMeasureCount
  \repeat unfold 7 {
    c'4 d' e' f'
  }
  \stopMeasureCount
}

```



The image displays the musical notation for the 'Seven Steps' exercise, organized into three staves. The first staff contains measures 1 through 7, the second staff contains measures 8 through 14, and the third staff contains measures 15 through 21. Each measure is numbered above the staff. The notation consists of eighth and sixteenth notes, with rests, all in a single melodic line. The key signature is one flat (B-flat), and the time signature is common time (C). The exercise is presented in a clear, black-and-white format suitable for educational purposes.

Measure spanners are an alternate way to print annotated brackets. As opposed to horizontal brackets, they extend between two bar lines rather than two notes. The text is displayed in the center of the bracket.

```
\layout {
  \context {
    \Staff
    \consists Measure_spanner_engraver
  }
}

<<

\new Staff \relative c'' {
  \key d \minor
  R1*2
  \tweak text "Answer"
  \startMeasureSpanner
  \tuplet 3/2 8 {
    a16[ b c] d[ c b]   c[ d e] f[ e d]
  }
  e8 a gis g
  fis f e d~ d c b e
  \stopMeasureSpanner
}
```

```

}
\new Staff \relative c' {
  \key d \minor
  \tweak text "Subject"
    \tweak direction #DOWN
    \startMeasureSpanner
  \tuplet 3/2 8 {
    d16[ e f] g[ f e] f[ g a] bes[ a g]
  }
  a8 d cis c
  b bes a g~ g f e a
  \stopMeasureSpanner
  \tweak text "Counter-subject"
    \tweak direction #DOWN
    \startMeasureSpanner
  f8 e a r r16 b, c d e fis g e
  a gis a b c fis, b a gis e a4 g8
  \stopMeasureSpanner
}
>>

```

Subject

Answer

Counter-subject

## Positioning fingering indications precisely

The semi-automatic positioning of fingering within a chords works usually well, but if one of the indications needs to be positioned more precisely the following tweak may be used. This is particularly useful for correcting the positioning when intervals of a second are involved.

```

\markup \with-true-dimensions % work around a cropping issue
\score {
  \relative c' {
    \set fingeringOrientations = #'(left)
    <c-1 d-2 a'-5>4
    <c-1 d-\tweak extra-offset #'(0 . 0.2)-2 a'-5>

    \set fingeringOrientations = #'(down)
    <c-1 d-2 a'-5>

```

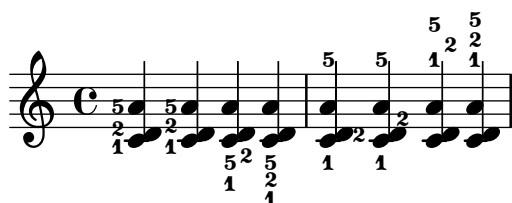
```

<c-\tweak extra-offset #'(0 . -1.1)-1
  d-\tweak extra-offset #'(-1.2 . -1.8)-2 a'-5> |

\set fingeringOrientations = #'(down right up)
<c-1 d-\tweak extra-offset #'(-0.3 . 0)-2 a'-5>4
<c-1 d-\tweak extra-offset #'(-1 . 1.2)-2 a'-5>

\set fingeringOrientations = #'(up)
<c-1 d-\tweak extra-offset #'(0 . 1.1)-2
  a'-\tweak extra-offset #'(0 . 1)-5>
<c-1 d-\tweak extra-offset #'(-1.2 . 1.5)-2
  a'-\tweak extra-offset #'(0 . 1.4)-5> |
}
}

```



## Positioning text markups inside slurs

Text markups need to have the `outside-staff-priority` property set to `#f` 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
}

```



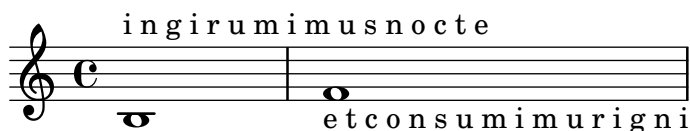
## Printing text from right to left

It is possible to print text from right to left in a markup object, as demonstrated here.

```

{
  b1^\markup {
    \line { i n g i r u m i m u s n o c t e }
  }
  f'_\markup {
    \override #'(text-direction . -1)
    \line { i n g i r u m i m u s n o c t e }
  }
}

```

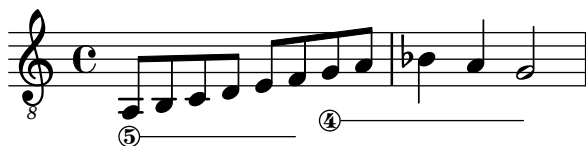


## String number extender lines

Make an extender line for string number indications, showing that a series of notes is supposed to be played all on the same string.

```
stringNumberSpanner =
  \#(define-music-function (StringNumber) (string?)
    \#{
      \override TextSpanner.style = #'solid
      \override TextSpanner.font-size = #-5
      \override TextSpanner.bound-details.left.stencil-align-dir-y = #CENTER
      \override TextSpanner.bound-details.left.text =
        \markup { \circle \number $StringNumber }
    \#})

\relative c {
  \clef "treble_8"
  \textSpannerDown
  \stringNumberSpanner "5" a8\startTextSpan b c d
    e f\stopTextSpan \stringNumberSpanner "4" g\startTextSpan a |
  bes4 a g2\stopTextSpan
}
```



## Using the whiteout property

Any graphical object can be printed over a white background to mask parts of objects that lie beneath. This can be useful to improve the appearance of collisions in complex situations when repositioning objects is impractical. It is necessary to explicitly set the layer property to control which objects are masked by the white background.

In this example the collision of the tie with the time signature is improved by masking out the part of the tie that crosses the time signature, setting the whiteout property of TimeSignature. To do this, TimeSignature is moved to a layer above Tie, which is left in the default layer 1, and StaffSymbol is moved to a layer above TimeSignature so it is not masked.

```
{
  \override Score.StaffSymbol.layer = 4
  \override Staff.TimeSignature.layer = 3
  b'2 b'~
  \once \override Staff.TimeSignature.whiteout = ##t
  \time 3/4
  b' r4
}
```



## 8 Text

See also Section “Text” in *Notation Reference*.

### Adding markups in a tablature

By default, markups are not displayed in a tablature.

To make them appear, revert the `stencil` property of the `TextScript` grob in the `TabStaff` context.

```
high = { r4 r8 <g c'> q r8 r4 }
low = { c4 r4 c8 r8 g,8 b, }
pulse = { s8^"1" s^"&" s^"2" s^"&" s^"3" s^"&" s^"4" s^"&" }
```

```
\score {
  \new TabStaff {
    \repeat unfold 2 << \high \\\ \low \\\ \pulse >>
  }
  \layout {
    \context {
      \TabStaff
      \clef moderntab
      \revert TextScript.stencil
      \override TextScript.font-series = #'bold
      \override TextScript.font-size = #-2
      \override TextScript.color = #red
    }
    \context {
      \Score
      proportionalNotationDuration = #1/8
    }
  }
}
```

	1	&	2	&	3	&	4	&	1	&	2	&	3	&	4	&
<b>T</b>					1	-	1						1	-	1	
<b>A</b>					0	-	0						0	-	0	
<b>B</b>	3				3			2	3				3			2
							3								3	

### Adding the current date to a score

With a little Scheme code, the current date can easily be added to a score.

```
\paper { tagline = ##f }
```

```
% first, define a variable to hold the formatted date:
date = #(strftime "%d-%m-%Y" (localtime (current-time)))
```

```
% use it in the title block:
```

```
\header {
  title = "Including the date!"
  subtitle = \date
}
```

```
\score {
  \relative c' {
    c4 c c c
  }
}
% and use it in a \markup block:
\markup {
  \date
}
```

## Including the date!

17-01-2026



17-01-2026

## Adjusting vertical spacing of lyrics

This snippet shows how to bring the lyrics line closer to the staff.

*% Default layout:*

```
<<
  \new Staff \new Voice = melody \relative c' {
    c4 d e f | g4 f e d | c1
  }
  \new Lyrics \lyricsto melody { aa aa aa aa aa aa aa aa aa }

  \new Staff \new Voice = melody \relative c' {
    c4 d e f | g4 f e d | c1
  }

  % Reducing the minimum space below the staff and above the lyrics:
  \new Lyrics \with {
    \override VerticalAxisGroup.nonstaff-relatedstaff-spacing =
      #'((basic-distance . 1))
  }
  \lyricsto melody { aa aa aa aa aa aa aa aa aa }
>>
```



## Aligning and centering instrument names

The horizontal alignment of instrument names is tweaked by changing the `self-alignment-X` property of the `InstrumentName` grob (usually in the `Staff` context). The `\layout` variables

`indent` and `short-indent` define the space in which the instrument names are aligned before the first and the following systems, respectively.

```
\paper {
  left-margin = 3\cm
}

\new StaffGroup <<
  \new Staff \with {
    \override InstrumentName.self-alignment-X = #LEFT
    instrumentName = \markup \left-column { "Left aligned"
                                             "instrument name" }

    shortInstrumentName = "Left"
  } {
    c''1 \break c''1
  }


  \new Staff \with {
    \override InstrumentName.self-alignment-X = #CENTER
    instrumentName = \markup \center-column { Centered
                                             "instrument name" }

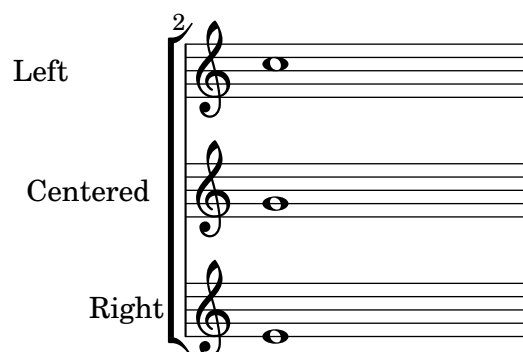
    shortInstrumentName = "Centered"
  } {
    g'1 g'1
  }

  \new Staff \with {
    \override InstrumentName.self-alignment-X = #RIGHT
    instrumentName = \markup \right-column { "Right aligned"
                                             "instrument name" }

    shortInstrumentName = "Right"
  } {
    e'1 e'1
  }
>>

\layout {
  indent = 4\cm
  short-indent = 2\cm
  line-width = 6.5\cm
}
```

Left aligned instrument name	
Centered instrument name	
Right aligned instrument name	

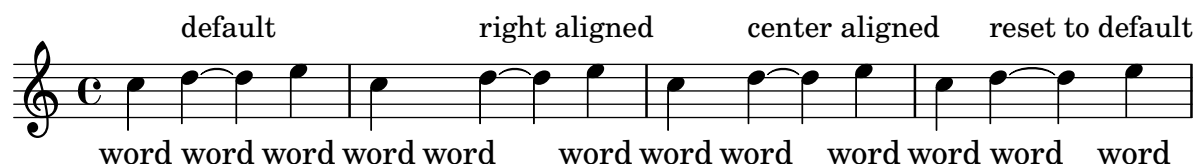


## Aligning syllables with melisma

By default, lyrics syllables that start a melisma are left-aligned on their note. The alignment can be altered using the `lyricMelismaAlignment` property.

```
<<
\new Staff {
  \new Voice = "vocal" \relative c' {
    \override TextScript.staff-padding = #2
    c d~\markup default d e
    c d~\markup "right aligned" d e
    c d~\markup "center aligned" d e
    c d~\markup "reset to default" d e
  }
}
\new Lyrics \lyricsto "vocal" {
  word word word
  \set lyricMelismaAlignment = #RIGHT
  word word word
  \set lyricMelismaAlignment = #CENTER
  word word word
  \unset lyricMelismaAlignment
  word word word
}
>>

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



## Aligning text marks to notes

By default, `TextMark` objects are aligned to so-called `NonMusicalPaperColumn` grobs, like the left edge of the staff or a bar line. They can be aligned to a note instead by setting the `non-musical` property to `#f`.

```
\layout {
  line-length = 80\mm
}
```



```

}

{
  \textMark "mark a" c'1 |
  \textMark "mark b" c'1 |
  \break
  \override Score.TextMark.non-musical = ##f
  \textMark "mark c" c'1 |
  \textMark "mark d" c'1 |
}

```



## Blanking staff lines using the `\whiteout` command

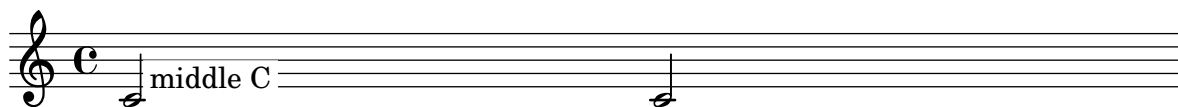
The `\whiteout` command underlays a markup with a white box. Since staff lines are in a lower layer than most other grobs, this white box will not overlap any other grob.

```

\layout {
  ragged-right = ##f
}

\relative c' {
  \override TextScript.extra-offset = #'(2 . 4)
  c2-\markup { \whiteout \pad-markup #0.5 "middle C" } c
}

```



## Center text below hairpin dynamics

This example provides a function to typeset a hairpin (de)crescendo with some additional text below it, such as “molto” or “poco”. The added text will change the direction according to the direction of the hairpin. The Hairpin is aligned to a `DynamicText` grob.

The example also illustrates how to modify the way an object is normally printed, using some Scheme code.

```

hairpinWithCenteredText =
#(define-music-function (text) (markup?)
  #{
    \once \override Voice.Hairpin.after-line-breaking =
      #(lambda (grob)
        (let* ((stencil (ly:hairpin::print grob))
              (par-y (ly:grob-parent grob Y))
              (dir (ly:grob-property par-y 'direction)))

```

```

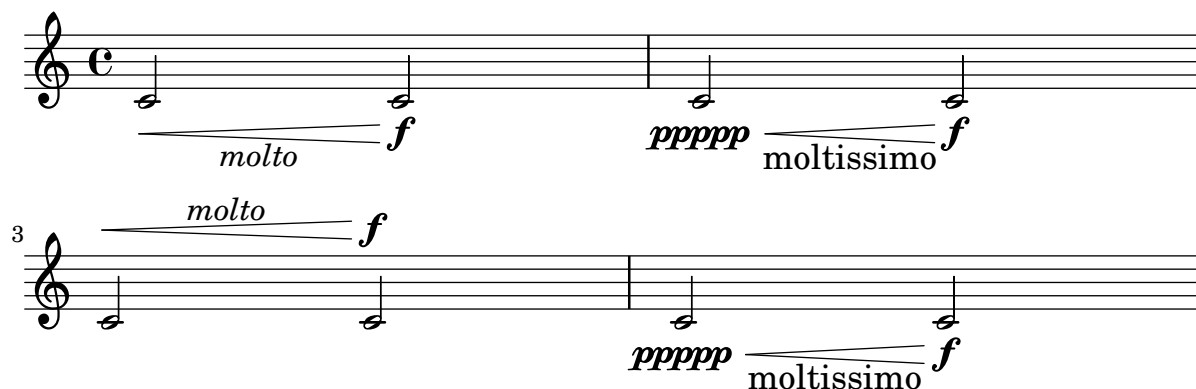
(staff-line-thickness
  (ly:output-def-lookup (ly:grob-layout grob)
                        'line-thickness))
(new-stencil
  (ly:stencil-aligned-to
    (ly:stencil-combine-at-edge
      (ly:stencil-aligned-to stencil X CENTER)
      Y dir
      (ly:stencil-aligned-to
        (grob-interpret-markup
          grob
          (make-fontsize-markup
            (magnification->font-size
              (+ (ly:staff-symbol-staff-space grob)
                (/ staff-line-thickness 2)))
            text))
        X CENTER))
      X LEFT))
  (staff-space (ly:output-def-lookup
    (ly:grob-layout grob) 'staff-space))
  (par-x (ly:grob-parent grob X))
  (dyn-text (grob::has-interface par-x
                                'dynamic-text-interface))

  (dyn-text-stencil-x-length
    (if dyn-text
      (interval-length
        (ly:stencil-extent
          (ly:grob-property par-x 'stencil) X))
      0))
  (x-shift
    (if dyn-text (- (+ staff-space dyn-text-stencil-x-length)
                    (* 0.5 staff-line-thickness))
      0)))
(ly:grob-set-property! grob 'Y-offset 0)
(ly:grob-set-property! grob
  'stencil (ly:stencil-translate-axis
    new-stencil
    x-shift X))))
#})

hairpinMolto = \hairpinWithCenteredText \markup { \italic molto }
hairpinMore = \hairpinWithCenteredText \markup { \larger moltissimo }

\relative c' {
  \hairpinMolto c2\< c\f
  \hairpinMore c2\ppppp\< c\f
  \break
  \hairpinMolto c2^\< c\f
  \hairpinMore c2\ppppp\< c\f
}

```



## Changing 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`.

Short text is especially useful when a brief ottava is used.

```
{
  c'2
  \ottava 1
  \set Staff.ottavation = "8"
  c''2
  \ottava 0
  c'1
  \ottava 1
  \set Staff.ottavation = "Text"
  c''1
}
```



## Changing the default text font family

The default font families for text can be overridden.

```
%{
You may have to install additional fonts.
```

```
Red Hat Fedora
```

```
dejavu-fonts-all
```

```
Debian GNU/Linux, Ubuntu
```

```
fonts-dejavu-core
```

```
fonts-dejavu-extra
```

```
%}
```

```
\paper {
  %{
    run
```

```

    lilypond -dshow-available-fonts
    to show all fonts available in the process log.
  %}
  property-defaults.fonts.serif = "DejaVu Serif"
  property-defaults.fonts.sans = "DejaVu Sans"
  property-defaults.fonts.typewriter = "DejaVu Sans Mono"
}

{
  g'''4^\markup {
    DejaVu Serif: \bold bold
                  \italic italic
                  \italic \bold { bold italic }
  }
  g4_\markup {
    \override #'(font-family . sans) {
      DejaVu Sans: \bold bold
                  \italic italic
                  \italic \bold { bold italic }
    }
  }
  g''2^\markup {
    \override #'(font-family . typewriter) {
      DejaVu Sans Mono: \bold bold
                      \italic italic
                      \italic \bold { bold italic }
    }
  }
}

```



## Combining dynamics with markup texts

Some dynamics may involve text indications (such as “*più f*” or “*p subito*”). These can be produced using a `\markup` block; the resulting object behaves like a `TextScript` grob.

See also “Combining dynamics with markup texts (2)”.

```

piuF = \markup { \italic più \dynamic f }

\markup \with-true-dimensions % work around a cropping issue
\score {
  \relative c'' {
    c2\f c-\piuF
  }
}

```



## Combining dynamics with markup texts (2)

Some dynamics may involve text indications (such as “*più f*” or “*p subito*”). These can be produced using the `make-dynamic-script` Scheme function; the resulting object behaves like a `DynamicText` grob.

See also “Combining dynamics with markup texts”.

```
piuF = #(make-dynamic-script
        #{ \markup { \normal-text \italic più \dynamic f } #})
```

```
\score {
  \relative c'' {
    c2\f c\piuF
  }
}
```



## Combining two parts on the same staff

The part combiner tool (i.e., the `\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 them 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.

```
musicUp = \relative c'' {
  \time 4/4
  a4 c4.( g8) a4 |
  g4 e' g,( a8 b) |
  c b a2.
}
```

```
musicDown = \relative c'' {
  g4 e4.( d8) c4 |
  r2 g'4( f8 e) |
  d2 \stemDown a
}
```

```
\score {
  <<
    \new Staff \with {
      instrumentName = "standard polyphony"
    } << \musicUp \\\musicDown >>
  >>
```

```

\new Staff \with {
  instrumentName =
    \markup { \typewriter "\\partCombine" without text}
  printPartCombineTexts = ##f
} \partCombine \musicUp \musicDown

\new Staff \with {
  instrumentName =
    \markup { \typewriter "\\partCombine" with text}
} \partCombine \musicUp \musicDown
>>

\layout {
  indent = 6.0\cm
  \context {
    \Score
    % Setting this to a large value avoids a bar line at the
    % beginning that would connect the three staves otherwise.
    \override SystemStartBar.collapse-height = 30
  }
}

```

standard polyphony	
\partCombine without text	
\partCombine with text	

## Creating “real” parenthesized dynamics

Although the easiest way to add parentheses to a dynamic mark is to use a `\markup` block, this method has a downside: the created objects behave like text markups and not like dynamics.

However, it is possible to create a similar object using the equivalent Scheme code (as described in the Notation Reference), combined with the `make-dynamic-script` function. This way, the markup is regarded as a dynamic and therefore remains compatible with commands such as `\dynamicUp` or `\dynamicDown`.

```

paren =
#(define-event-function (dyn) (ly:event?)
  (make-dynamic-script
    #{ \markup \concat {
      \normal-text \italic \fontsize #2 (
        \pad-x #0.2 #(ly:music-property dyn 'text)
        \normal-text \italic \fontsize #2 )
    }
  )

```

```
#}))
```

```
\relative c' ' {
  c4\paren\f c c \dynamicUp c\paren\p
}
```



## Creating text spanners

The `\startTextSpan` and `\stopTextSpan` commands allow the creation of text spanners as easily as pedal indications or octavations. Override some properties of the `TextSpanner` object to modify its output.

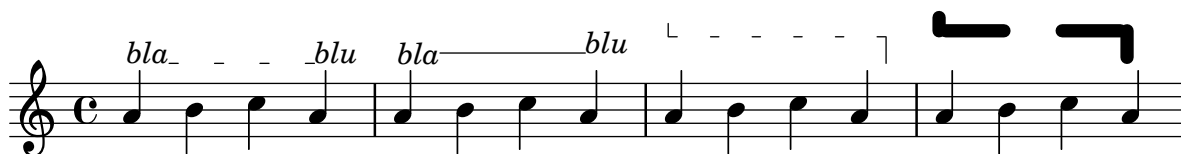
```
\paper { ragged-right = ##f }
```

```
\relative c' ' {
  \override TextSpanner.bound-details.left.text = #"bla"
  \override TextSpanner.bound-details.right.text = #"blu"
  a4 \startTextSpan
  b4 c
  a4 \stopTextSpan

  \override TextSpanner.style = #'line
  \once \override TextSpanner.bound-details.left.stencil-align-dir-y = #CENTER
  a4 \startTextSpan
  b4 c
  a4 \stopTextSpan

  \override TextSpanner.style = #'dashed-line
  \override TextSpanner.bound-details.left.text =
    \markup { \draw-line #'(0 . 1) }
  \override TextSpanner.bound-details.right.text =
    \markup { \draw-line #'(0 . -2) }
  \once \override TextSpanner.bound-details.right.padding = #-2
  a4 \startTextSpan
  b4 c
  a4 \stopTextSpan

  \override TextSpanner.dash-period = #10
  \override TextSpanner.dash-fraction = #0.5
  \override TextSpanner.thickness = #10
  a4 \startTextSpan
  b4 c
  a4 \stopTextSpan
}
```



## Demonstrating all \header fields

A demonstration of all header fields that LilyPond defines by default. Thanks to setting `print-all-headers` to `#t`, much more fields as usual are displayed, indicating the hierarchy of `\header` blocks.

```
\paper {
  #(set-paper-size "a6" 'landscape)
  print-all-headers = #t
}


\book {
  \header {
    title = "title"
    subtitle = "subtitle"
    composer = "composer"
    arranger = "arranger"
    instrument = "instrument"
    meter = "meter"
    opus = "opus"
    piece = "piece"
    poet = "poet"
    copyright = "copyright"
    tagline = "tagline"
  }

  \bookpart {
    \score {
      \relative c'' { c1 | c | c | c }

      \header {
        title = "localtitle"
        subtitle = "localsubtitle"
        composer = "localcomposer"
        arranger = "localarranger"
        instrument = "localinstrument"
        meter = "localmeter"
        opus = "localopus"
        piece = "localpiece"
        poet = "localpoet"
        copyright = "localcopyright"
        tagline = "localtagline"
      }
    }
  }
}
```



	<b>title</b>	
	<b>subtitle</b>	
poet	<b>instrument</b>	composer
meter		arranger
	<b>localtitle</b>	
	<b>localsubtitle</b>	
localpoet	<b>localinstrument</b>	localcomposer
localmeter		localarranger
localpiece		localopus

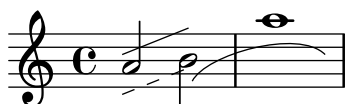
	copyright	
	tagline	

## Embedding native PostScript in a \markup block

PostScript code can be directly inserted inside a \markup block.

In general it is recommended to use LilyPond's native graphical markup commands like \polygon instead, which can be used with all LilyPond backends.

```
\relative c' ' {
  a2-\markup \postscript "0 3 moveto
                        5 2 rlineto
                        stroke"
  -\markup \postscript "[1 1] 0 setdash
                        0 0 moveto
                        5 2 rlineto
                        stroke"
  b2-\markup \postscript "1 1 moveto
                        0 0 1 2 8 4 10 2 rcurveto
                        stroke"
  a'1
}
```

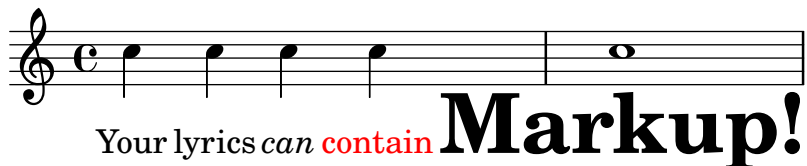


## Formatting lyrics syllables

The \markup command can be used in \lyricmode blocks to format individual syllables in lyrics.

```
mel = \relative c' ' { c4 c c c c1 }
lyr = \lyricmode {
  Your lyrics \markup { \italic can }
  \markup { \with-color #red contain }
  \markup { \fontsize #8 \bold Markup! }
}
```

```
<<
  \new Voice = "melody" \mel
  \new Lyrics \lyricsto "melody" \lyr
>>
```



## How to put ties between syllables in lyrics

This can be achieved by separating those syllables by tildes.

```
\lyrics {
  wa~o~a
}
```

wa o a

## Lyrics alignment

Horizontal alignment for lyrics can be set by overriding the `self-alignment-X` property of the `LyricText` object. Value `-1` means left-aligned, `0` centered, and `1` right-aligned. Alternatively, you can use the Scheme values `LEFT`, `CENTER`, and `RIGHT` instead of numbers. Other numeric values are possible, too – don't forget to add the `'#` Scheme prefix for negative numbers!

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

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

```
\addlyrics {
  \once \override LyricText.self-alignment-X = #LEFT
  "left-aligned"
  \once \override LyricText.self-alignment-X = #CENTER
  "centered"
  \once \override LyricText.self-alignment-X = 1
  "right-aligned"
  \once \override LyricText.self-alignment-X = #-1.5
  "very right"
}
```



## Markup list

Text that can spread over pages is entered with the `\markuplist` command. The `\paragraph` markup command defined in the snippet indents its argument before calling `\justified-lines`.

```
#(set-default-paper-size "a6" 'landscape)

\paper {
  line-width = 11\cm
  tagline = ##f
}

#(define-markup-list-command (paragraph layout props args) (markup-list?)
  (interpret-markup-list layout props
    (make-justified-lines-markup-list (cons (make-hspace-markup 2) args))))

\book { % for correct rendering in the PDF documentation
  % Candide, Voltaire
  \markuplist {
    \override-lines #'(baseline-skip . 2.5) {
      \paragraph {
        Il y avait en Westphalie, dans le château de M. le baron de
        Thunder-ten-tronckh, un jeune garçon à qui la nature avait donné
        les mœurs les plus douces. Sa physionomie annonçait son âme.
        Il avait le jugement assez droit, avec l'esprit le plus
        \concat { simple \hspace #.3 ; }
        c'est, je crois, pour cette raison qu'on le nommait Candide. Les
        anciens domestiques de la maison soupçonnaient qu'il était fils
        de la sœur de monsieur le baron et d'un bon et honnête
        gentilhomme du voisinage, que cette demoiselle ne voulut jamais
        épouser parce qu'il n'avait pu prouver que soixante et onze
        quartiers, et que le reste de son arbre généalogique avait été
        perdu par l'injure du temps.
      }
      \vspace #.3
      \paragraph {
        Monsieur le baron était un des plus puissants seigneurs de la
        Westphalie, car son château avait une porte et des fenêtres. Sa
        grande salle même était ornée d'une tapisserie. Tous les chiens
        de ses basses-cours composaient une meute dans le
        \concat { besoin \hspace #.3 ; }
        ses palefreniers étaient ses
        \concat { piqueurs \hspace #.3 ; }
        le vicaire du village était
        son grand-aumônier. Ils l'appelaient tous monseigneur, et ils
        riaient quand il faisait des contes.
      }
    }
  }
}
```

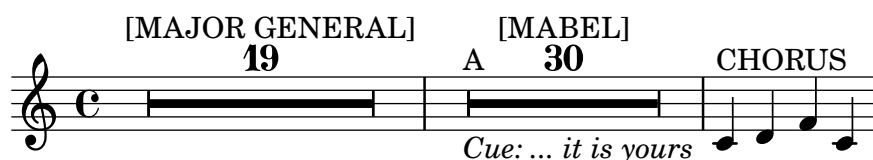
Il y avait en Westphalie, dans le château de M. le baron de Thunder-ten-tronckh, un jeune garçon à qui la nature avait donné les mœurs les plus douces. Sa physionomie annonçait son âme. Il avait le jugement assez droit, avec l'esprit le plus simple; c'est, je crois, pour cette raison qu'on le nommait Candide. Les anciens domestiques de la maison soupçonnaient qu'il était fils de la sœur de monsieur le baron et d'un bon et honnête gentilhomme du voisinage, que cette demoiselle ne voulut jamais épouser parce qu'il n'avait pu prouver que soixante et onze quartiers, et que le reste de son arbre généalogique avait été perdu par l'injure du temps.

Monsieur le baron était un des plus puissants seigneurs de la Westphalie, car son château avait une porte et des fenêtres. Sa grande salle même était ornée d'une tapisserie. Tous les chiens de ses basses-cours composaient une meute dans le besoin; ses palefreniers étaient ses piqueurs; le vicaire du village était son grand-aumônier. Ils l'appelaient tous monseigneur, et ils riaient quand il faisait des contes.

## Multi-measure rest markup

Markups attached to a multi-measure rest will be centered above or below it. Long markups attached to multi-measure rests do not cause the measure to expand. To expand a multi-measure rest to fit the markup, use an empty chord with an attached markup before the multi-measure rest. Text attached to a spacer rest in this way is left-aligned to the position where the note would be placed in the measure, but if the measure length is determined by the length of the text, the text will appear to be centered.

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



## Of the ubiquity of markup objects

Text objects are entered either as simple strings between double quotes or as `\markup` blocks that can accept a variety of advanced text formatting and graphical enhancements.

As such, markup blocks may be used:

- in any `TextScript` object (attached to notes with `-`, `^` or `_`),
- any `TextMark` introduced with the `\textMark` keyword, or `\textEndMark` command, or other similar objects such as `MetronomeMark` introduced with `\tempo`,

- as standalone markup blocks, entered at the top level outside of any `\score` block,
- in any definition inside the `\header` block (e.g. `title`, `subtitle`, `composer`) or in some variables defined inside the `\paper` block such as `evenHeaderMarkup` for page numbers.

`\markup` may additionally be used for lyrics, in chord names, and as dynamics. In fact, it is possible to use `\markup` to customize the appearance of virtually any object, as demonstrated in this example using various methods.

*%% Thanks to Aaron Hill [https://lists.gnu.org/archive/html/lilypond-user/2019-01/msg00437.h](https://lists.gnu.org/archive/html/lilypond-user/2019-01/msg00437.html)*

```
\paper {
  paper-width = 8\cm paper-height = 8\cm
}

\header {
  title = \markup "Header"
  tagline = \markup "(tagline)"
}

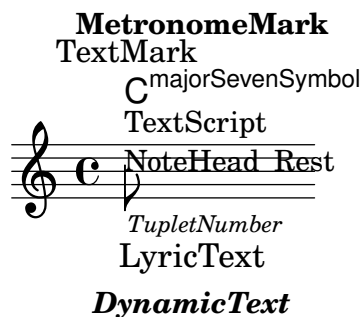
\markup "Top-level markup"

dyn = #(make-dynamic-script #{ \markup \serif "DynamicText" #})

\score {
  <<
    \new ChordNames
    \with { majorSevenSymbol = \markup "majorSevenSymbol" }
    \chordmode { c1:maj7 }
    \new Staff {
      \tempo \markup "MetronomeMark"
      \textMark "TextMark"
      \once \override TupletNumber.text = \markup "TupletNumber"
      \tuplet 3/2 {
        \once \override NoteHead.stencil = #ly:text-interface::print
        \once \override NoteHead.text = \markup \lower #0.5 "NoteHead"
        c' '8^\markup "TextScript"
        \once \override Rest.stencil = #(lambda (grob)
          (grob-interpret-markup grob #{
            \markup "Rest"
          #}))
        r4
      }
    }
    \new Lyrics \lyricmode { \markup "LyricText" 1 }
    \new Dynamics { s1\dyn }
  >>
}
```

## Header

Top-level markup



## Outputting the version number

It is possible to print the version number of LilyPond in markup.

```
\markup { Processed with LilyPond version #(lilypond-version) }
```

Processed with LilyPond version 2.25.32

## Piano template with centered lyrics

Instead of having a full staff for the melody and lyrics, lyrics can be centered between the staves of a piano staff.

```
upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

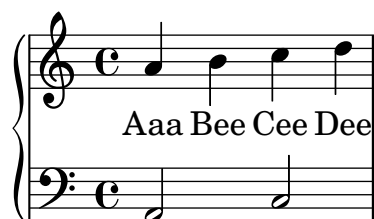
  a4 b c d
}

lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  a2 c
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

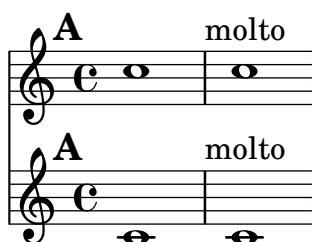
\score {
  \new PianoStaff <<
    \new Staff = upper { \new Voice = "singer" \upper }
    \new Lyrics \lyricsto "singer" \text
    \new Staff = lower { \lower }
  >>
  \layout { }
  \midi { }
}
```



## Printing marks on every staff

Although rehearsal and text marks are normally only printed above the topmost staff, they may also be printed on every staff.

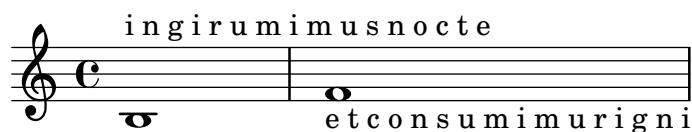
```
\score {
  <<
    \new Staff { \mark \default c''1 \textMark "molto" c'' }
    \new Staff { \mark \default c'1 \textMark "molto" c' }
  >>
  \layout {
    \context {
      \Score
      \remove Mark_engraver
      \remove Text_mark_engraver
      \remove Staff_collecting_engraver
    }
    \context {
      \Staff
      \consists Mark_engraver
      \consists Text_mark_engraver
      \consists Staff_collecting_engraver
    }
  }
}
```



## Printing text from right to left

It is possible to print text from right to left in a markup object, as demonstrated here.

```
{
  b1~\markup {
    \line { i n g i r u m i m u s n o c t e }
  }
  f'_~\markup {
    \override #'(text-direction . -1)
    \line { i n g i r u m i m u s n o c t e }
  }
}
```



## Putting lyrics inside the staff

Lyrics can be moved vertically to place them inside the staff. The lyrics are moved with `\override LyricText.extra-offset = #'(0 . dy)`, and there are similar commands to move the extenders and hyphens. A good value for *dy* must be found by trial and error.

```
<<
\new Staff <<
  \new Voice = "voc" \relative c' { \stemDown a bes c8 b c4 }
>>
\new Lyrics \with {
  \override LyricText.extra-offset = #'(0 . 8.6)
  \override LyricExtender.extra-offset = #'(0 . 8.6)
  \override LyricHyphen.extra-offset = #'(0 . 8.6)
} \lyricsto "voc" { La la -- la __ _ la }
>>
```



## Stand-alone two-column markup

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

```
\markup {
  \fill-line {
    \hspace #1
    \column {
      \line { 0 sacrum convivium }
      \line { in quo Christus sumitur, }
      \line { recolitur memoria passionis ejus, }
      \line { mens impletur gratia, }
      \line { futurae gloriae nobis pignus datur. }
      \line { Amen. }
    }
    \hspace #2
    \column \italic {
      \line { 0 sacred feast }
      \line { in which Christ is received, }
      \line { the memory of His Passion is renewed, }
      \line { the mind is filled with grace, }
      \line { and a pledge of future glory is given to us. }
      \line { Amen. }
    }
  }
  \hspace #1
}
```



O sacrum convivium  
in quo Christus sumitur,  
recolitur memoria passionis ejus,  
mens impletur gratia,  
futuræ gloriæ nobis pignus datur.  
Amen.

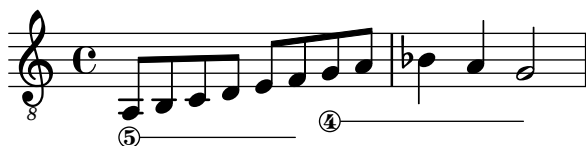
*O sacred feast  
in which Christ is received,  
the memory of His Passion is renewed,  
the mind is filled with grace,  
and a pledge of future glory is given to us.  
Amen.*

## String number extender lines

Make an extender line for string number indications, showing that a series of notes is supposed to be played all on the same string.

```
stringNumberSpanner =
  #(define-music-function (StringNumber) (string?)
    #{
      \override TextSpanner.style = #'solid
      \override TextSpanner.font-size = #-5
      \override TextSpanner.bound-details.left.stencil-align-dir-y = #CENTER
      \override TextSpanner.bound-details.left.text =
        \markup { \circle \number $StringNumber }
    #})

\relative c {
  \clef "treble_8"
  \textSpannerDown
  \stringNumberSpanner "5" a8\startTextSpan b c d
    e f\stopTextSpan \stringNumberSpanner "4" g\startTextSpan a |
  bes4 a g2\stopTextSpan
}
```



## Three-sided box

This example shows how to add a markup command to get a three-sided box around some text (or other markup).

```
% New command to add a three-sided box, with sides north, west, and south.
% Based on the `box-stencil` command defined in `scm/stencil.scm`.
% Note that ";" is used to comment a line in Scheme.
#(define-public (NWS-box-stencil stencil thickness padding)
  "Add a box around STENCIL, producing a new stencil."
  (let* ((x-ext (interval-widen (ly:stencil-extent stencil X) padding))
        (y-ext (interval-widen (ly:stencil-extent stencil Y) padding))
        (y-rule (make-filled-box-stencil (cons 0 thickness) y-ext))
        (x-rule (make-filled-box-stencil
                  (interval-widen x-ext thickness) (cons 0 thickness))))
    ;; (set! stencil (ly:stencil-combine-at-edge stencil X 1 y-rule padding))
    (set! stencil (ly:stencil-combine-at-edge stencil X LEFT y-rule padding))
    (set! stencil (ly:stencil-combine-at-edge stencil Y UP x-rule 0.0))
    (set! stencil (ly:stencil-combine-at-edge stencil Y DOWN x-rule 0.0))
```

```

stencil))

% The corresponding markup command, based on the '\box' command defined
% in `scm/define-markup-commands.scm`.
#(define-markup-command (NWS-box layout props arg) (markup?)
  #:properties ((thickness 0.1) (font-size 0) (box-padding 0.2))
  "Draw a box round ARG.

```

Look at THICKNESS, BOX-PADDING, and FONT-SIZE properties to determine line thickness and padding around the markup."

```

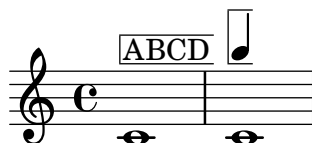
  (let ((pad (* (magstep font-size) box-padding))
        (m (interpret-markup layout props arg)))
    (NWS-box-stencil m thickness pad)))

```

```

\relative c' {
  c1~\markup { \NWS-box ABCD }
  c1~\markup { \NWS-box \note {4} #1.0 }
}

```



## UTF-8

Various scripts may be used for texts (like titles and lyrics) by entering them in UTF-8 encoding, and using a Pango based backend. Depending on the fonts installed, this fragment will render Bulgarian (Cyrillic), Hebrew, Japanese and Portuguese.

```

%{
You may have to install additional fonts.

```

*Red Hat Fedora*

```

linux-libertine-fonts (Latin, Cyrillic, Hebrew)
google-noto-serif-jp-fonts (Japanese)

```

*Debian GNU/Linux, Ubuntu*

```

fonts-linuxlibertine (Latin, Cyrillic, Hebrew)
fonts-noto-cjk (Japanese)
}%

```

% 'Linux Libertine' fonts also contain Cyrillic and Hebrew glyphs.

```

\paper {
  property-defaults.fonts.serif =
    "Linux Libertine O, Noto Serif CJK JP, Noto Serif JP"
}

```

```

bulgarian = \lyricmode {
  Жълтата дюля беше щастлива, че пухът, който цъфна, замръзна като гьон.
}

```

```

}

hebrew = \lyricmode {
.
}

japanese = \lyricmode {

}

% "a nice song for you"
portuguese = \lyricmode {
à vo -- cê uma can -- ção le -- gal
}

\relative c' {
c2 d
e2 f
g2 f
e2 d
}

\addlyrics { \bulgarian }
\addlyrics { \hebrew }
\addlyrics { \japanese }
\addlyrics { \portuguese }

```

The image shows a musical score for a vocal ensemble. It consists of two staves, each with four measures. The first staff has a treble clef and a common time signature (C). The notes are quarter notes. The lyrics are written below the staff, with each line of lyrics corresponding to a measure. The lyrics are in four languages: Bulgarian, Hebrew, Japanese, and Portuguese. The second staff is similar, with a treble clef and a common time signature (C). The notes are quarter notes. The lyrics are written below the staff, with each line of lyrics corresponding to a measure. The lyrics are in four languages: Bulgarian, Hebrew, Japanese, and Portuguese.

Measure	Bulgarian	Hebrew	Japanese	Portuguese
1	Жълтата	הַ	いろはにほへど	à
2	дюля	כִּי	ちりぬるを	vo -
3	беше	סָתָם	わがよたれぞ	- cê
4	щастлива,	לְשִׁמּוֹעַ	つねならむ	uma
5	че	וְאֵיךְ	うゐのおくや	can -
6	пухът,	תִּנְצָה	まけふこえて	- ção
7	който	קִרְפָּה	あさきゆめみじ	le -
8	цъфна,	עֵץ	糸ひもせず	- gal

## Vocal ensemble template with lyrics aligned below and above the staves

This template is basically the same as the simple “Vocal ensemble template”, with the exception that here all the lyrics lines are placed using `alignAboveContext` and `alignBelowContext`.

```

global = {
\key c \major

```

```

\time 4/4
}

sopMusic = \relative c'' {
  c4 c c8[( b)] c4
}
sopWords = \lyricmode {
  hi hi hi hi
}

altoMusic = \relative c' {
  e4 f d e
}
altoWords = \lyricmode {
  ha ha ha ha
}

tenorMusic = \relative c' {
  g4 a f g
}
tenorWords = \lyricmode {
  hu hu hu hu
}

bassMusic = \relative c {
  c4 c g c
}
bassWords = \lyricmode {
  ho ho ho ho
}

\score {
  \new ChoirStaff <<
    \new Staff = "women" <<
      \new Voice = "sopranos" { \voiceOne << \global \sopMusic >> }
      \new Voice = "altos" { \voiceTwo << \global \altoMusic >> }
    >>
    \new Lyrics \with { alignAboveContext = "women" }
      \lyricsto "sopranos" \sopWords
    \new Lyrics \with { alignBelowContext = "women" }
      \lyricsto "altos" \altoWords
    % we could remove the line about this with the line below, since
    % we want the alto lyrics to be below the alto Voice anyway.
    % \new Lyrics \lyricsto "altos" \altoWords

    \new Staff = "men" <<
      \clef bass
      \new Voice = "tenors" { \voiceOne << \global \tenorMusic >> }
      \new Voice = "basses" { \voiceTwo << \global \bassMusic >> }
    >>
    \new Lyrics \with { alignAboveContext = "men" }
      \lyricsto "tenors" \tenorWords

```

```

\new Lyrics \with { alignBelowContext = "men" }
  \lyricsto "basses" \bassWords
% again, we could replace the line above this with the line below.
% \new Lyrics \lyricsto "basses" \bassWords
>>
}

```



## Volta text markup using repeatCommands

Though volte are best specified using `\repeat volta`, the context property `repeatCommands` must be used in cases where the volta text needs more advanced formatting with `\markup`.

Since `repeatCommands` takes a list, the simplest method of including markup is to use an identifier for the text and embed it in the command list using the Scheme syntax `#(list (list 'volta textIdentifier))`. Start- and end-repeat commands can be added as separate list elements:

```
voltaAdLib = \markup { \volta-number { 1. 2. 3... } \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))
}

```





**Specialist notation**





## 9 Vocal music

See also Section “Vocal music” in *Notation Reference*.

### Adding ambitus per voice

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

```
\new Staff <<
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c' {
    \override Ambitus.X-offset = 2.0
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c' {
    \voiceTwo
    es4 f g as
    b1
  }
}>>
```



### Adding indicators to staves which get split after a break

This snippet defines the commands `\splitStaffBarLine`, `\convUpStaffBarLine`, and `\convDownStaffBarLine`. These add arrows at a bar line to denote that several voices sharing a staff will each continue on a staff of their own in the next system, or that voices split in this way recombine.

Note that the implementation in this snippet draws dimensionless arrows into the right margin. For normal printing, this doesn't cause problems. However, it is necessary to increase the bounding box horizontally if you render the code as an image to avoid cropping, as demonstrated below.

```
#(define-markup-command (arrow-at-angle layout props angle-deg length fill)
  (number? number? boolean?)
  (let* ((PI-OVER-180 (/ (atan 1 1) 34))
        (degrees->radians (lambda (degrees) (* degrees PI-OVER-180)))
        (angle-rad (degrees->radians angle-deg))
        (target-x (* length (cos angle-rad)))
        (target-y (* length (sin angle-rad))))
    (interpret-markup layout props
      (markup
        #:translate (cons (/ target-x 2) (/ target-y 2))
        #:rotate angle-deg
        #:translate (cons (/ length -2) 0))
```

```

      #:concat (:#draw-line (cons length 0)
        #:arrow-head X RIGHT fill))))))

splitStaffBarLineMarkup = \markup \with-dimensions #'(0 . 0) #'(0 . 0) {
  \combine
  \arrow-at-angle #45 #(sqrt 8) ##t
  \arrow-at-angle #-45 #(sqrt 8) ##t
}

splitStaffBarLine = {
  \once \override Staff.BarLine.stencil =
  #(\lambda (grob)
    (ly:stencil-combine-at-edge
      (ly:bar-line::print grob)
      X RIGHT
      (grob-interpret-markup grob splitStaffBarLineMarkup)
      0))
  \break
}

convDownStaffBarLine = {
  \once \override Staff.BarLine.stencil =
  #(\lambda (grob)
    (ly:stencil-combine-at-edge
      (ly:bar-line::print grob)
      X RIGHT
      (grob-interpret-markup grob #{
        \markup\with-dimensions #'(0 . 0) #'(0 . 0) {
          \translate #'(0 . -.13)\arrow-at-angle #-45 #(sqrt 8) ##t
        }#}))
    0))
  \break
}

convUpStaffBarLine = {
  \once \override Staff.BarLine.stencil =
  #(\lambda (grob)
    (ly:stencil-combine-at-edge
      (ly:bar-line::print grob)
      X RIGHT
      (grob-interpret-markup grob #{
        \markup\with-dimensions #'(0 . 0) #'(0 . 0) {
          \translate #'(0 . .14)\arrow-at-angle #45 #(sqrt 8) ##t
        }#}))
    0))
  \break
}

\paper {
  indent = 10\mm
  short-indent = 10\mm
  line-width = 8\cm
}

```

```

}

separateSopranos = {
  \set Staff.instrumentName = "AI AII"
  \set Staff.shortInstrumentName = "AI AII"
  \splitStaffBarLine
  \change Staff = "up"
}

convSopranos = {
  \convDownStaffBarLine
  \change Staff = "shared"
  \set Staff.instrumentName = "S A"
  \set Staff.shortInstrumentName = "S A"
}

sI = {
  \voiceOne
  \repeat unfold 4 f''2
  \separateSopranos
  \repeat unfold 4 g''2
  \convSopranos
  \repeat unfold 4 c''2
}

sII = {
  s1*2
  \voiceTwo
  \change Staff = "up"
  \repeat unfold 4 d''2
}

aI = {
  \voiceTwo
  \repeat unfold 4 a'2
  \voiceOne
  \repeat unfold 4 b'2
  \convUpStaffBarLine
  \voiceTwo
  \repeat unfold 4 g'2
}

aII = {
  s1*2
  \voiceTwo
  \repeat unfold 4 g'2
}

ten = {
  \voiceOne
  \repeat unfold 4 c'2
  \repeat unfold 4 d'2
  \repeat unfold 4 c'2
}

bas = {
  \voiceTwo
  \repeat unfold 4 f2

```

```

\repeat unfold 4 g2
\repeat unfold 4 c2
}

\markup \pad-x #3 % avoid cropping
\score {
  <<
    \new ChoirStaff <<
      \new Staff = up \with {
        instrumentName = "SI SII"
        shortInstrumentName = "SI SII"
      } {
        s1*4
      }

      \new Staff = shared \with {
        instrumentName = "S A"
        shortInstrumentName = "S A"
      } <<
        \new Voice = sopI \sI
        \new Voice = sopII \sII
        \new Voice = altI \aI
        \new Voice = altII \aII
      >>
      \new Lyrics \with {
        alignBelowContext = up
      }
      \lyricsto sopII { e f g h }
      \new Lyrics \lyricsto altI { a b c d e f g h i j k l }

      \new Staff = men \with {
        instrumentName = "T B"
        shortInstrumentName = "T B"
      } <<
        \clef F
        \new Voice = ten \ten
        \new Voice = bas \bas
      >>
      \new Lyrics \lyricsto bas { a b c d e f g h i j k l }
    >>
  >>

  \layout {
    \context {
      \Staff \RemoveEmptyStaves
      \override VerticalAxisGroup.remove-first = ##t
    }
  }
}

```

The image displays three musical systems, each with two staves. The first system shows Soprano (S A) and Tenor/Bass (T B) parts with lyrics 'a b c d'. The second system shows Soprano I (SI SII) and Alto I (AI AII) parts with lyrics 'e f g h'. The third system shows Soprano (S A) and Tenor/Bass (T B) parts with lyrics 'i j k l'. Each staff contains four notes, and the lyrics are aligned below the notes.

## Adding orchestral cues to a vocal score

This snippet shows one approach to simplify adding many orchestral cues to the piano reduction in a vocal score. The music function `\cueWhile` takes four arguments: the music from which the cue is to be taken, as defined by `\addQuote`, the name to be inserted before the cue notes, then either UP or DOWN to specify either `\voiceOne` with the name above the staff or `\voiceTwo` with the name below the staff, and finally the piano music in parallel with which the cue notes are to appear. The name of the cued instrument is positioned to the left of the cued notes. Many passages can be cued, but they cannot overlap each other in time.

```
cueWhile =
#(define-music-function
  (instrument name dir music)
  (string? string? ly:dir? ly:music?)
  #{
    \cueDuring $instrument #dir {
      \once \override TextScript.self-alignment-X = #RIGHT
      \once \override TextScript.direction = $dir
```

```

        <>-\markup { \tiny #name }
        $music
    }
    #})

flute = \relative c'' {
    \transposition c'
    s4 s4 e g
}
\addQuote "flute" { \flute }

clarinet = \relative c' {
    \transposition bes
    fis4 d d c
}
\addQuote "clarinet" { \clarinet }

singer = \relative c'' { c4. g8 g4 bes4 }
words = \lyricmode { here's the lyr -- ics }

pianoRH = \relative c'' {
    \transposition c'
    \cueWhile "clarinet" "Clar." #DOWN { c4. g8 }
    \cueWhile "flute" "Flute" #UP { g4 bes4 }
}
pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  <<
    \new Staff {
      \new Voice = "singer" {
        \singer
      }
    }
    \new Lyrics {
      \lyricsto "singer"
      \words
    }
    \new PianoStaff <<
      \new Staff {
        \new Voice {
          \pianoRH
        }
      }
      \new Staff {
        \clef "bass"
        \pianoLH
      }
    >>
  >>
}

```



## Adjusting vertical spacing of lyrics

This snippet shows how to bring the lyrics line closer to the staff.

*% Default layout:*

```
<<
  \new Staff \new Voice = melody \relative c' {
    c4 d e f | g4 f e d | c1
  }
  \new Lyrics \lyricsto melody { aa aa aa aa aa aa aa aa }

  \new Staff \new Voice = melody \relative c' {
    c4 d e f | g4 f e d | c1
  }

  % Reducing the minimum space below the staff and above the lyrics:
  \new Lyrics \with {
    \override VerticalAxisGroup.nonstaff-relatedstaff-spacing =
      #'((basic-distance . 1))
  }
  \lyricsto melody { aa aa aa aa aa aa aa aa }
>>
```



## Aligning syllables with melisma

By default, lyrics syllables that start a melisma are left-aligned on their note. The alignment can be altered using the `lyricMelismaAlignment` property.

```
<<
  \new Staff {
    \new Voice = "vocal" \relative c' {
      \override TextScript.staff-padding = #2
      c d~\markup default d e
      c d~\markup "right aligned" d e
      c d~\markup "center aligned" d e
    }
  }
>>
```

```

      c d~\markup "reset to default" d e
    }
  }
  \new Lyrics \lyricsto "vocal" {
    word word word
    \set lyricMelismaAlignment = #RIGHT
    word word word
    \set lyricMelismaAlignment = #CENTER
    word word word
    \unset lyricMelismaAlignment
    word word word
  }
>>

\layout {
  ragged-right = ##f
}

```



## Ambitus

Ambitus indicate pitch ranges for voices.

Accidentals only show up if they are not part of the key signature. AmbitusNoteHead grobs also have ledger lines.

```

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

<<
\new Staff {
  \relative c' {
    \time 2/4
    c4 f'
  }
}
\new Staff {
  \relative c' {
    \time 2/4
    \key d \major
    cis4 as'
  }
}
>>

```





## Ambitus after key signature

By default, ambitus are positioned at the left of the clef. The `\ambitusAfter` function allows for changing this placement. Syntax is `\ambitusAfter grob-interface`; see Graphical Object Interfaces (<http://lilypond.org/doc/v2.24/Documentation/internals/graphical-object-interfaces>) for a list of possible values for *grob-interface*.

A common use case is printing the ambitus between key signature and time signature.

```
\new Staff \with {
  \consists Ambitus_engraver
} \relative {
  \ambitusAfter key-signature
  \key d \major
  es'8 g bes cis d2
}
```



## 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
  }
}>>
```



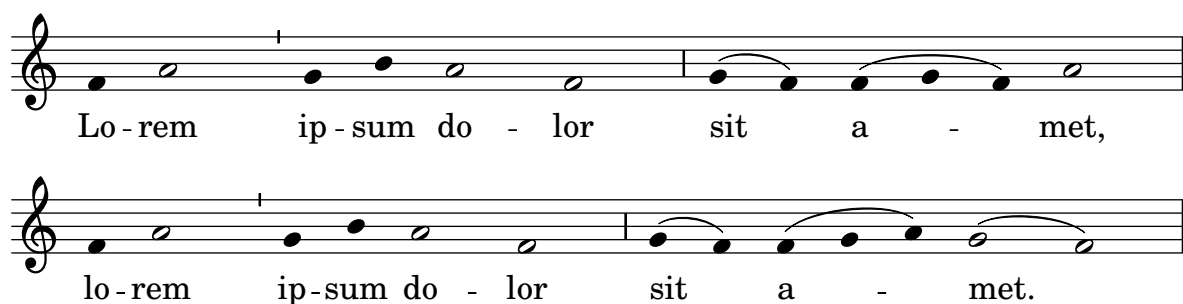
## Ancient notation template – modern transcription of Gregorian music

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.

```
chant = \relative c' {
  \set Score.timing = ##f
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g f) a2 \finalis \break
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g a) g2( f) \finalis
}

verba = \lyricmode {
  Lo -- rem ip -- sum do -- lor sit a -- met,
  lo -- rem ip -- sum do -- lor sit a -- met.
}

\score {
  \new GregorianTranscriptionStaff <<
    \new GregorianTranscriptionVoice = "melody" \chant
    \new GregorianTranscriptionLyrics = "one" \lyricsto melody \verba
  >>
}
```



## Anglican psalm template

This template shows one way of setting out an Anglican psalm chant. It also shows how the verses may be added as stand-alone text under the music. The two verses are coded in different styles to demonstrate more possibilities.

```
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
}

dot = \markup {
    \raise #0.7 \musicglyph "dots.dot"
}

tick = \markup {
    \raise #1 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}

% Use markup to center the chant on the page
\markup \fill-line {
    \score { % centered
        \new ChoirStaff <<
            \new Staff <<
                \global
                \clef "treble"
                \new Voice = "Soprano" <<
                    \voiceOne
                    \SopranoMusic
                >>
                \new Voice = "Alto" <<
                    \voiceTwo
                    \AltoMusic
                >>
            >>
        >>

        \new Staff <<
            \clef "bass"
            \global
            \new Voice = "Tenor" <<
                \voiceOne
                \TenorMusic
            >>
            \new Voice = "Bass" <<
                \voiceTwo
                \BassMusic
            >>
        >>
    >>
}

```

```

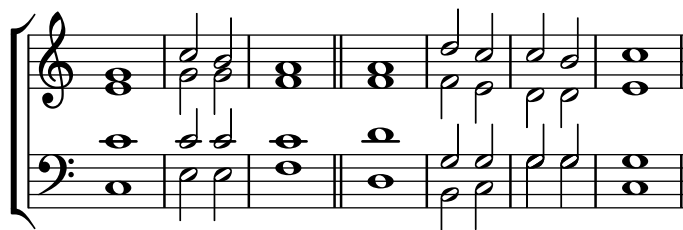
\layout {
  \context {
    \Score
    \override SpacingSpanner.base-shortest-duration =
      \musicLength 2
  }
  \context {
    \Staff
    \remove "Time_signature_engraver"
  }
}
} % End score
} % End markup

\markup \fill-line {
  \left-column {
    \null \null \null
    \line { \fontsize #5 0
      \fontsize #3 come
      let us \bold sing | unto \dot the | Lord : let }
    \line { us heartily \concat { re \bold joyce }
      in the | strength of | our }
    \line { sal | vation. }

    \null

    \line { \hspace #2.5 8. Today if ye will hear his voice * }
    \line { \concat { \bold hard en }
      \tick not your \tick hearts : as in the pro- }
    \line { vocation * and as in the \bold day of tempt- \tick }
    \line { -ation \tick in the \tick wilderness. }
  }
}

```



**O** come let us **sing** | unto • the | Lord : let  
us heartily **rejoyce** in the | strength of | our  
sal | vation.

8. Today if ye will hear his voice \*  
**harden** ' not your ' hearts : as in the pro-  
vocation \* and as in the **day** of tempt- '  
-ation ' in the ' wilderness.

## Arranging separate lyrics on a single line

Sometimes you may want to put lyrics for different performers on a single line: where there is rapidly alternating text, for example. This snippet shows how it can be done with adjusting the `nonstaff-nonstaff-spacing` property of the `VerticalAxisGroup` grob.

```
\layout {
  \context {
    \Lyrics
    \override VerticalAxisGroup
      .nonstaff-nonstaff-spacing
      .minimum-distance = ##f
  }
}

aliceSings = \markup { \smallCaps "Alice" }
eveSings = \markup { \smallCaps "Eve" }

<<
\new Staff <<
  \new Voice = "alice" {
    f'4^\aliceSings g' r2 |
    s1 |
    f'4^\aliceSings g' r2 |
    s1 | \break
    % ...

    \voiceOne
    s2 a'8^\aliceSings a' b'4 |
    \oneVoice
    g'1
  }
  \new Voice = "eve" {
    s1 |
    a'2^\eveSings g' |
    s1 |
    a'2^\eveSings g'
    % ...

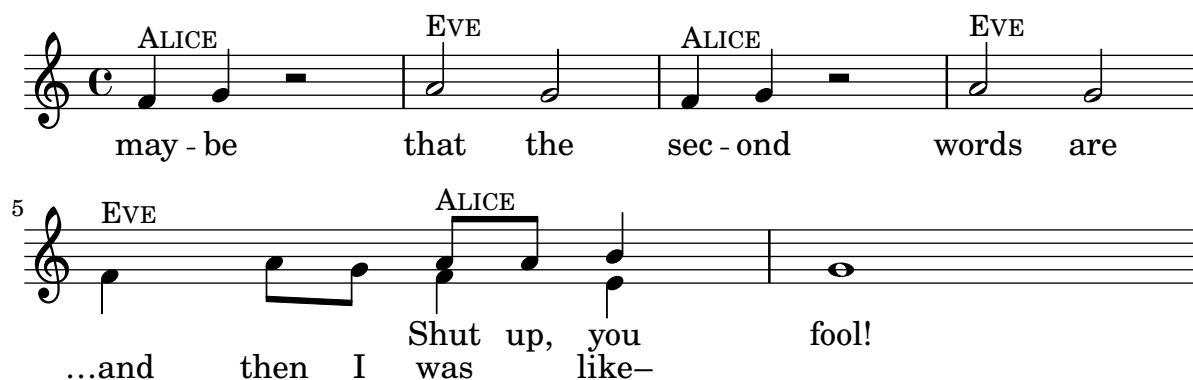
    \voiceTwo
    f'4^\eveSings a'8 g' f'4 e' |
    \oneVoice
    s1
  }
}
>>

\new Lyrics \lyricsto "alice" {
  may -- be
  sec -- ond
  % ...
  Shut up, you fool!
}
```

```

\new Lyrics \lyricsto "eve" {
  that the
  words are
  % ...
  ...and then I was like--
}
>>

```



## Changing stanza fonts

Fonts can be changed independently for each stanza, including the font used for printing the stanza number.

```

%{
You may have to install additional fonts.

```

Red Hat Fedora

```

dejavu-fonts-all

```

Debian GNU/Linux, Ubuntu

```

fonts-dejavu-core
fonts-dejavu-extra
%}

```

```

\relative c' ' {
  \time 3/4
  g2 e4
  a2 f4
  g2.
}
\addlyrics {
  \set stanza = #"1. "
  Hi, my name is Bert.
}
\addlyrics {
  \override StanzaNumber.fonts.serif = "DejaVu Sans"
  \set stanza = #"2. "
  \override LyricText.font-family = #'typewriter
  Oh, ché -- ri, je t'aime
}

```



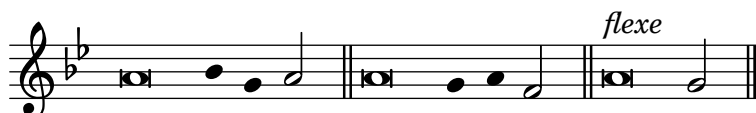
1. Hi, my name is Bert.
2. Oh, ché-ri, jet'aime

## Chant or psalm notation

This form of notation is used for psalm chant, where verses aren't always of the same length.

```
stemOff = \hide Staff.Stem
stemOn  = \undo \stemOff

\score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \key g \minor
    \cadenzaOn
    \stemOff a'\breve bes'4 g'4
    \stemOn a'2 \section
    \stemOff a'\breve g'4 a'4
    \stemOn f'2 \section
    \stemOff a'\breve^\markup { \italic flexe }
    \stemOn g'2 \fine
  }
}
```



## Forcing hyphens to be shown

If LilyPond does not think there is space for a hyphen, it will be omitted. This behaviour can be overridden with the minimum-distance property of LyricHyphen.

```
\relative c'' {
  c32 c c c
  c32 c c c
  c32 c c c
  c32 c c c
}

\addlyrics {
  syl -- lab word word
  \override LyricHyphen.minimum-distance = #1.0
  syl -- lab word word
  \override LyricHyphen.minimum-distance = #2.0
  syl -- lab word word
  \revert LyricHyphen.minimum-distance
  syl -- lab word word
}
```



## Formatting lyrics syllables

The `\markup` command can be used in `\lyricmode` blocks to format individual syllables in lyrics.

```
mel = \relative c'' { c4 c c c c1 }
lyr = \lyricmode {
  Your lyrics \markup { \italic can }
  \markup { \with-color #red contain }
  \markup { \fontsize #8 \bold Markup! }
}
```

```
<<
  \new Voice = "melody" \mel
  \new Lyrics \lyricsto "melody" \lyr
>>
```



## How to put ties between syllables in lyrics

This can be achieved by separating those syllables by tildes.

```
\lyrics {
  wa~o~a
}

wa o a
```

## Hymn template

This code shows one way of setting out a hymn tune where 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 \caesura \break
  s2 | s1 | s2 \breathe s2 | s1 | s2 \fine
}

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 {
  \left-column {
    "This is line one of the first verse"
    "This is line two of the same"
    \null
    "And here's line one of the second verse"
    "And the next line of the same"
  }
}

\layout {
  \context {
    \Score
    caesuraType = #'((bar-line . "||"))
    fineBarType = "||"
  }
}

\paper { % Start paper block
  indent = 0 % don't indent first system
  line-width = 130 % shorten line length to suit music
  tagline = ##f % Don't print tag line, can be removed
} % End paper block

```



This is line one of the first verse  
 This is line two of the same

And here's line one of the second verse  
 And the next line of the same

## Lyrics alignment

Horizontal alignment for lyrics can be set by overriding the `self-alignment-X` property of the `LyricText` object. Value `-1` means left-aligned, `0` centered, and `1` right-aligned. Alternatively, you can use the Scheme values `LEFT`, `CENTER`, and `RIGHT` instead of numbers. Other numeric values are possible, too – don't forget to add the `'#` Scheme prefix for negative numbers!

```

\layout {
  ragged-right = ##f

```

```

}

\relative c' {
  c1 c c c
}

\addlyrics {
  \once \override LyricText.self-alignment-X = #LEFT
  "left-aligned"
  \once \override LyricText.self-alignment-X = #CENTER
  "centered"
  \once \override LyricText.self-alignment-X = 1
  "right-aligned"
  \once \override LyricText.self-alignment-X = #-1.5
  "very right"
}

```



## Marking notes of spoken parts with a cross on the stem (Sprechstimme)

This example shows how to put crosses on stems. Mark the beginning of a spoken section with the command `\speakOn` and end it with `\speakOff`.

```

speakOn = \override Stem.stencil =
  #(\lambda (grob)
    (let* ((x-parent (ly:grob-parent grob X))
      (is-rest? (ly:grob? (ly:grob-object x-parent 'rest))))
      (if is-rest?
        empty-stencil
        (ly:stencil-combine-at-edge
          (ly:stem::print grob)
          Y
          (- (ly:grob-property grob 'direction))
          (grob-interpret-markup
            grob
            (markup #:center-align #:fontsize -4
              #:musicglyph "noteheads.s2cross"))
          -1.7))))))

speakOff = \revert Stem.stencil

\new Staff {
  \relative c' {
    a4 b a c
    \speakOn
    g4 f r g8 a
    b4 r r8 d e4
    \speakOff
  }
}

```

```

    c4 a g f
  }
}

```



## Orchestra, choir and piano template

This template demonstrates the use of nested `StaffGroup` and `GrandStaff` contexts to subgroup instruments of the same type together, and a way to use `\transpose` so that variables hold music for transposing instruments at concert pitch.

```

#(set-global-staff-size 17)

```

```

\paper {
  indent = 3.0\cm % add space for instrumentName
  short-indent = 1.5\cm % add less space for shortInstrumentName
}

```

```

fluteMusic = \relative c' { \key g \major g'1 b }

```

```

% Pitches as written on a manuscript for Clarinet in A
% are transposed to concert pitch.

```

```

clarinetMusic = \transpose c' a
  \relative c'' { \key bes \major bes1 d }

```

```

trumpetMusic = \relative c { \key g \major g''1 b }

```

```

% Key signature is often omitted for horns

```

```

hornMusic = \transpose c' f
  \relative c { d'1 fis }

```

```

percussionMusic = \relative c { \key g \major g1 b }

```

```

sopranoMusic = \relative c'' { \key g \major g'1 b }

```

```

sopranoLyrics = \lyricmode { Lyr -- ics }

```

```

altoIMusic = \relative c' { \key g \major g'1 b }

```

```

altoILyrics = \sopranoLyrics

```

```

altoIIMusic = \relative c' { \key g \major g'1 b }

```

```

altoIILyrics = \lyricmode { Ah -- ah }

```

```

tenorMusic = \relative c' { \clef "treble_8" \key g \major g1 b }

```

```

tenorLyrics = \sopranoLyrics

```

```

pianoRHMus = \relative c { \key g \major g''1 b }

```

```

pianoLHMus = \relative c { \clef bass \key g \major g1 b }

```

```

violinIMusic = \relative c' { \key g \major g'1 b }

```

```

violinIIMusic = \relative c' { \key g \major g'1 b }

```

```

violaMusic = \relative c { \clef alto \key g \major g'1 b }

celloMusic = \relative c { \clef bass \key g \major g1 b }

bassMusic = \relative c { \clef "bass_8" \key g \major g,1 b }

\book {
  \score {
    <<
    \new StaffGroup = "StaffGroup_woodwinds" <<
    \new Staff = "Staff_flute" \with { instrumentName = "Flute" }
    \fluteMusic

    \new Staff = "Staff_clarinet" \with {
      instrumentName = \markup { \concat { "Clarinet in B" \flat } }
    }
    % Declare that written Middle C in the music
    % to follow sounds a concert B flat, for
    % output using sounded pitches such as MIDI.
    %\transposition bes

    % Print music for a B-flat clarinet
    \transpose bes c' \clarinetMusic
    >>

    \new StaffGroup = "StaffGroup_brass" <<
    \new Staff = "Staff_hornI" \with {
      instrumentName = "Horn in F"
    }
    % \transposition f
    \transpose f c' \hornMusic

    \new Staff = "Staff_trumpet" \with {
      instrumentName = "Trumpet in C"
    }
    \trumpetMusic
    >>

    \new RhythmicStaff = "RhythmicStaff_percussion" \with {
      instrumentName = "Percussion"
    }
    \percussionMusic

    \new PianoStaff \with {
      instrumentName = "Piano"
    } <<
    \new Staff { \pianoRHMusical }
    \new Staff { \pianoLHMusical }
    >>

    \new ChoirStaff = "ChoirStaff_choir" <<
    \new Staff = "Staff_soprano" \with {

```

```

    instrumentName = "Soprano"
}
\new Voice = "soprano" \sopranoMusic
\new Lyrics \lyricsto "soprano" { \sopranoLyrics }

\new GrandStaff = "GrandStaff_altos" \with {
  \accepts Lyrics
} <<
  \new Staff = "Staff_altoI" \with {
    instrumentName = "Alto I"
  }
  \new Voice = "altoI"
  \altoIMusic
  \new Lyrics \lyricsto "altoI" { \altoILyrics }
  \new Staff = "Staff_altoII" \with {
    instrumentName = "Alto II"
  }
  \new Voice = "altoII"
  \altoIIMusic
  \new Lyrics \lyricsto "altoII" { \altoIILyrics }
>>

\new Staff = "Staff_tenor" \with {
  instrumentName = "Tenor"
}
\new Voice = "tenor" \tenorMusic
\new Lyrics \lyricsto "tenor" { \tenorLyrics }
>>

\new StaffGroup = "StaffGroup_strings" <<
  \new GrandStaff = "GrandStaff_violins" <<
    \new Staff = "Staff_violinI" \with {
      instrumentName = "Violin I"
    }
    \violinIMusic
    \new Staff = "Staff_violinII" \with {
      instrumentName = "Violin II"
    }
    \violinIIMusic
  >>
>>

\new Staff = "Staff_viola" \with {
  instrumentName = "Viola"
}
\violaMusic

\new Staff = "Staff_cello" \with {
  instrumentName = "Cello"
}
\celloMusic

\new Staff = "Staff_bass" \with {

```

```
        instrumentName = "Double Bass"  
    }  
    \bassMusic  
    >>  
    >>  
    }  
}
```

Flute

Clarinet in B $\flat$

Horn in F

Trumpet in C

Percussion

Piano

Soprano

Alto I

Alto II

Tenor

Violin I

Violin II

Viola

Cello

Double Bass

Lyr - ics

Lyr - ics

Ah - ah

Lyr - ics

8

8

Detailed description: This is a musical score for a vocal and instrumental ensemble, showing measures 8 and 9. The score is written for a variety of instruments and voices. The key signature is one sharp (F#), and the time signature is common time (C). The instruments listed on the left are Flute, Clarinet in B $\flat$ , Horn in F, Trumpet in C, Percussion, Piano, Soprano, Alto I, Alto II, Tenor, Violin I, Violin II, Viola, Cello, and Double Bass. The vocal parts (Soprano, Alto I, Alto II, Tenor) have lyrics written below them. The instrumental parts (Flute, Clarinet in B $\flat$ , Horn in F, Trumpet in C, Piano, Violin I, Violin II, Viola, Cello, Double Bass) are written on staves. The Percussion part is written on a single line. The score is divided into two measures, 8 and 9, by a double bar line. The music is written in a standard musical notation style, with notes, rests, and clefs. The vocal parts are written in a soprano, alto, and tenor clef. The instrumental parts are written in a standard staff clef. The lyrics are written in a simple, sans-serif font. The page number 276 is in the top right corner. The chapter title 'Chapter 9: Vocal music' is in the top left corner.



## Piano template with melody and lyrics

Here is a typical song format: one staff with the melody and lyrics, with piano accompaniment underneath.

```
melody = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

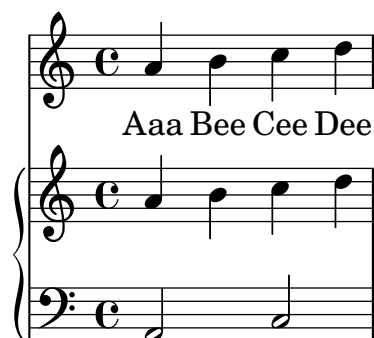
upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  a2 c
}

\score {
  <<
    \new Voice = "mel" { \autoBeamOff \melody }
    \new Lyrics \lyricsto mel \text
    \new PianoStaff <<
      \new Staff = "upper" \upper
      \new Staff = "lower" \lower
    >>
  >>
  \layout {
    \context { \Staff \RemoveEmptyStaves }
  }
  \midi { }
}
```



## Putting lyrics inside the staff

Lyrics can be moved vertically to place them inside the staff. The lyrics are moved with `\override LyricText.extra-offset = #'(0 . dy)`, and there are similar commands to move the extenders and hyphens. A good value for *dy* must be found by trial and error.

```
<<
\new Staff <<
  \new Voice = "voc" \relative c' { \stemDown a bes c8 b c4 }
>>
\new Lyrics \with {
  \override LyricText.extra-offset = #'(0 . 8.6)
  \override LyricExtender.extra-offset = #'(0 . 8.6)
  \override LyricHyphen.extra-offset = #'(0 . 8.6)
} \lyricsto "voc" { La la -- la _ _ la }
>>
```



## SATB choir template – four staves

This is a template for a SATB choir on four staves.

```
global = {
  \key c \major
  \time 4/4
  \dynamicUp
}
sopranonotes = \relative c'' {
  c2 \p \< d c d \f
}
sopranowords = \lyricmode { do do do do }
altonotes = \relative c'' {
  c2\p d c d
}
altowords = \lyricmode { re re re re }
tenornotes = {
  \clef "G_8"
  c2\mp d c d
}
```

```

tenorwords = \lyricmode { mi mi mi mi }
bassnotes = {
  \clef bass
  c2\mf d c d
}
basswords = \lyricmode { mi mi mi mi }

\score {
  \new ChoirStaff <<
    \new Staff <<
      \new Voice = "soprano" <<
        \global
        \sopranonotes
      >>
      \new Lyrics \lyricsto "soprano" \sopranowords
    >>
    \new Staff <<
      \new Voice = "alto" <<
        \global
        \altonotes
      >>
      \new Lyrics \lyricsto "alto" \altowords
    >>
    \new Staff <<
      \new Voice = "tenor" <<
        \global
        \tenornotes
      >>
      \new Lyrics \lyricsto "tenor" \tenorwords
    >>
    \new Staff <<
      \new Voice = "bass" <<
        \global
        \bassnotes
      >>
      \new Lyrics \lyricsto "bass" \basswords
    >>
  >>
}

```



## Single-staff template with notes, lyrics, and chords

This template allows the preparation of a song with melody, words, and chords.

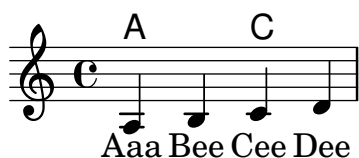
```
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

harmonies = \chordmode {
  a2 c
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \harmonies
    }
    \new Voice = "one" { \autoBeamOff \melody }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
  \midi { }
}
```



## Single-staff template with notes, lyrics, chords, and frets

Here is a simple lead sheet template with melody, lyrics, chords, and fret diagrams.

```

verseI = \lyricmode {
  \set stanza = #"1."
  This is the first verse
}

verseII = \lyricmode {
  \set stanza = #"2."
  This is the second verse.
}

theChords = \chordmode {
  % insert chords for chordnames and fretboards here
  c2 g4 c
}

staffMelody = \relative c' {
  \key c \major
  \clef treble
  % Type notes for melody here
  c4 d8 e f4 g
  \bar "|"
}

\score {
  <<
    \context ChordNames { \theChords }
    \context FretBoards { \theChords }
    \new Staff {
      \context Voice = "voiceMelody" { \staffMelody }
    }
    \new Lyrics = "lyricsI" {
      \lyricsto "voiceMelody" \verseI
    }
    \new Lyrics = "lyricsII" {
      \lyricsto "voiceMelody" \verseII
    }
  >>
  \layout { }
  \midi { }
}

```

The musical notation for the first verse is written on a single staff in treble clef with a common time signature (C). The melody consists of four notes: a quarter note on G4, an eighth note on A4, a quarter note on B4, and a half note on C5. Above the staff, three guitar chord diagrams are shown for the notes G, A, and B. The G chord is a barre at the 3rd fret (labeled '3fr') with the 2nd and 3rd strings open. The A chord is a barre at the 1st fret (labeled '1 0fr') with the 2nd and 3rd strings open. The B chord is a barre at the 3rd fret (labeled '3fr') with the 2nd and 3rd strings open. Below the staff, the lyrics are written in two lines: '1. This is the first verse' and '2. This is the second verse.'

## Single-staff template with notes and lyrics

This small template demonstrates a simple melody with lyrics. Cut and paste, add notes, then words for the lyrics. This example turns off automatic beaming, which is common for vocal parts. To use automatic beaming, change or comment out the relevant line.

```
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

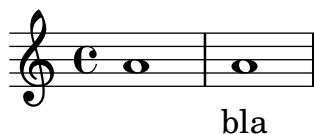
\score{
  <<
    \new Voice = "one" {
      \autoBeamOff
      \melody
    }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
  \midi { }
}
```

Aaa Bee Cee Dee

## Skips in lyric mode

The ‘s’ syntax for skips is only available in note mode and chord mode. In other situations, for example, when entering lyrics, using the `\skip` command is recommended.

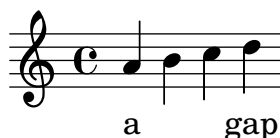
```
<<
  \relative c'' { a1 | a }
  \new Lyrics \lyricmode { \skip1 bla1 }
>>
```



## Skips in lyric mode (2)

Although 's' skips cannot be used in `\lyricmode` (it is taken to be a literal "s", not a space), double quotes (") or underscores ( ) are available.

```
<<
\relative c'' { a4 b c d }
\new Lyrics \lyricmode { a4 "" _ gap }
>>
```

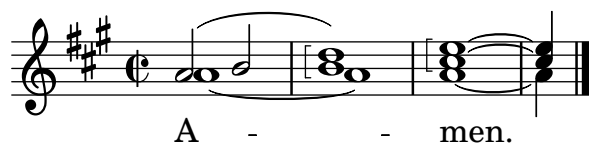


## Using `\arpeggioBracket` to make divisi more visible

The `\arpeggioBracket` command can be used to indicate the division of voices where there are no stems to provide the information. This is often seen in choral music.

```
\include "english.ly"

\score {
  \relative c'' {
    \key a \major
    \time 2/2
    <<
      \new Voice = "upper"
      <<
        { \voiceOne \arpeggioBracket
          a2( b2
            <b d>1\arpeggio)
            <cs e>\arpeggio ~
            <cs e>4
          }
        \addlyrics { \lyricmode { A -- men. } }
      >>
      \new Voice = "lower"
      { \voiceTwo
        a1 ~
        a
        a ~
        a4 \bar "|"
      }
    >>
  }
}
```



## Using tags to produce mensural and modern music from the same source

Using tags it is possible to produce both mensural and modern notation from the same music. In this snippet, a function `\menrest` is introduced, allowing mensural rests to be pitched as in the original, but with modern rests in the standard staff position.

Tags can also be used where other differences are needed: for example using “whole measure rests” (`R1`, `R\breve`, etc.) in modern music, but normal rests (`r1`, `r\breve`, etc.) in the mensural version. Converting mensural music to its modern equivalent is usually referred to as *transcription*.

The call `c4.\Be c8 c\Am` is the same as `c4.[ c8 c]`. However, it suppresses warnings if it starts on a note that can’t hold a beam but needs it anyway due to the use of `Completion_heads_engraver`.

[Note that the custos sticks out into the right margin and might be cut off if the LilyPond output gets cropped tightly. The use of `\with-true-dimensions` below avoids this.]

```
\layout {
  line-width = 150\mm
}

menrest = #(define-music-function (note) (ly:music?)
  #{
    \tag #'mens $(make-music 'RestEvent note)
    \tag #'mod $(make-music 'RestEvent note 'pitch '())
  #})

Be = \tag #'mod
  #(begin
    (ly:expect-warning (G_ "stem does not fit in beam"))
    (ly:expect-warning (G_ "beam was started here"))
    (make-span-event 'BeamEvent START))

Am = \tag #'mod ]

MenStyle = {
  \override Score.BarNumber.transparent = ##t
  \override Stem.neutral-direction = #up
  \omit Slur
  \omit Beam
}

finalis = \section

Music = \relative c'' {
  \key f \major
  g1 d'2 \menrest bes4 bes a2 \menrest r4 g4 fis4.
  fis8 fis4 fis g e f4.([ g8] a4[ g8 f]
    g2.\Be fis8 e\Am fis2) g\breve \finalis
}
```



```

MenLyr = \lyricmode {
  So farre, deere life, deare life,
  from thy bright beames ab- en- ted,
}
ModLyr = \lyricmode {
  So far, dear life, dear life,
  from your bright beams ab -- sen -- ted, __
}

\markup \with-true-dimensions % work around a cropping issue
\score {
  \keepWithTag #'mens {
    <<
      \new PetrucciStaff {
        \new PetrucciVoice = "Cantus" {
          \clef "petrucci-c1" \time 4/4 \MenStyle \Music
        }
      }
      \new Lyrics \lyricsto "Cantus" \MenLyr
    >>
  }
  \layout {
    \context {
      \PetrucciVoice
      % No longer necessary starting with version 2.25.23.
      \override Flag.style = #'mensural
    }
  }
}

\markup\vspace #1

\score {
  \keepWithTag #'mod {
    \new ChoirStaff <<
      \new Staff {
        \new Voice = "Sop" \with {
          \remove "Note_heads_engraver"
          \consists "Completion_heads_engraver"
          \remove "Rest_engraver"
          \consists "Completion_rest_engraver"
        } \shiftDurations 1 0 { \time 2/4 \autoBeamOff \Music }
      }
      \new Lyrics \lyricsto "Sop" \ModLyr
    >>
  }
}

```

So farre, deere life, deare life, from thy bright  
beames ab- fen- ted,

So far, dear life, dear life, from your bright  
beams ab - sen - - - ted,\_\_\_\_\_

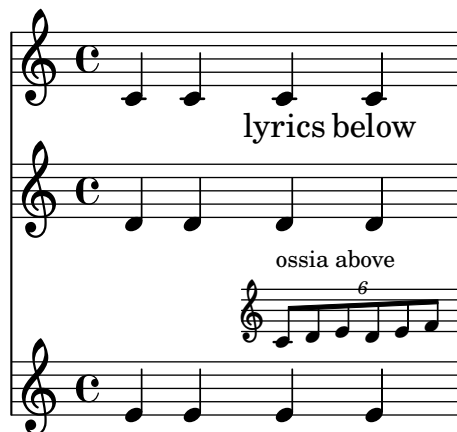
## Vertically aligning ossias and lyrics

This snippet demonstrates the use of the context properties `alignBelowContext` and `alignAboveContext` to control the positioning of lyrics and ossias.

```
\relative c' <<
  \new Staff = "1" { c4 c c c }
  \new Staff = "2" { d4 d d d }
  \new Staff = "3" { e4 e e e }

  { \skip 2
    <<
      \lyrics {
        \set alignBelowContext = "1"
        lyrics4 below
      }
      \new Staff \with {
        alignAboveContext = "3"
        fontSize = -2
        \override StaffSymbol.staff-space = #(magstep -2)
        \remove "Time_signature_engraver"
        \override VerticalAxisGroup.staff-staff-spacing =
          #'((minimum-distance . 0)
             (basic-distance . 0)
             (padding . 1))
      } {
        \tuplet 6/4 {
          \override TextScript.padding = 2
          c8["^"ossia above" d e d e f]
        }
      }
    }
  >>
```

```
}
>>
```



## Vertically aligning stanza numbers of different staves

It can happen that stanza numbers don't align vertically if the verses are attached to different staves. To fix that, override the `self-alignment-X` property of the `LyricText` grob.

```
\markup { default behavior }
```

```
<<
```

```
\new Staff { b b b b }
\lyrics {
  \set stanza = "3."
  a a a a
}
```

```
\new Staff { b b b b }
\lyrics {
  \set stanza = "1."
  aaaaaaaaaa a a a
}
```

```
\lyrics {
  \set stanza = "2."
  a a a a
}
```

```
>>
```

```
\markup \vspace #1
```

```
\markup {
  using \typewriter "self-alignment-X = #LEFT" }
```

```
<<
```

```
\new Staff { b b b b }
\new Lyrics \lyricmode {
  \set stanza = "3."
  a a a a
}
```

```
\new Staff { b b b b }
```

```

\new Lyrics \lyricmode {
  \set stanza = "1."
  \once \override LyricText.self-alignment-X = #LEFT
  aaaaaaaaaa a a a
}
\new Lyrics \lyricmode {
  \set stanza = "2."
  a a a a
}
>>

```

default behavior



using self-alignment-X = #LEFT



## Vertically centered common lyrics

In a vocal piece where there are several (two, four or more) lines of lyrics and common lyrics for all voices at some point, the common lyrics may be made to appear vertically centered, as shown in the following example:

```

dropLyrics = {
  \override LyricText.extra-offset = #'(0 . -4.5)
  \override LyricHyphen.extra-offset = #'(0 . -4.5)
  \override LyricExtender.extra-offset = #'(0 . -4.5)
  \override StanzaNumber.extra-offset = #'(0 . -4.5)
}

```

```

raiseLyrics = {
  \revert LyricText.extra-offset
  \revert LyricHyphen.extra-offset
  \revert LyricExtender.extra-offset
  \revert StanzaNumber.extra-offset
}

```

```

}

skipFour = \repeat unfold 4 { \skip 8 }

lyricsA = \lyricmode {
  The first verse has
  \dropLyrics
  \set stanza = #"  All:"
  the com -- mon __ words
  \raiseLyrics
  used in all four.
}

lyricsB = \lyricmode { In stan -- za two,   \skipFour al -- so ap -- pear. }

lyricsC = \lyricmode { By the third verse, \skipFour are get -- ting dull. }

lyricsD = \lyricmode { Last stan -- za, and \skipFour get used once more. }

melody = \relative c' {
  c4 d e f |
  g f e8( e f) d |
  c4 e d c |
}

\score {
  <<
    \new Voice = m \melody
    \new Lyrics \lyricsto m \lyricsA
    \new Lyrics \lyricsto m \lyricsB
    \new Lyrics \lyricsto m \lyricsC
    \new Lyrics \lyricsto m \lyricsD
  >>
}

```

The first verse has  
In stan - za two,  
By the third verse,  
Last stan - za, and

used in all four.  
al - so ap - pear.  
All: the common words  
are get - ting dull.  
get used once more.

## Vocal ensemble template

Here is a standard four-part SATB vocal score. With larger ensembles, it is often useful to include a section which is included in all parts. For example, the time signature and key signature are almost always the same for all parts. Like in the “Hymn template”, the four voices are regrouped on only two staves.

```

\paper {
  top-system-spacing.basic-distance = 10
  score-system-spacing.basic-distance = 20
}

```

```

    system-system-spacing.basic-distance = 20
    last-bottom-spacing.basic-distance = 10
}

global = {
  \key c \major
  \time 4/4
}

sopMusic = \relative {
  c''4 c c8[( b)] c4
}
sopWords = \lyricmode {
  hi hi hi hi
}

altoMusic = \relative {
  e'4 f d e
}
altoWords = \lyricmode {
  ha ha ha ha
}

tenorMusic = \relative {
  g4 a f g
}
tenorWords = \lyricmode {
  hu hu hu hu
}

bassMusic = \relative {
  c4 c g c
}
bassWords = \lyricmode {
  ho ho ho ho
}

\score {
  \new ChoirStaff <<
    \new Lyrics = "sopranos" \with {
      % this is needed for lyrics above a staff
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
    \new Staff = "women" <<
      \new Voice = "sopranos" {
        \voiceOne
        << \global \sopMusic >>
      }
      \new Voice = "altos" {
        \voiceTwo
        << \global \altoMusic >>
      }
    }
  }

```

```

>>
\new Lyrics = "altos"
\new Lyrics = "tenors" \with {
  % this is needed for lyrics above a staff
  \override VerticalAxisGroup.staff-affinity = #DOWN
}
\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" \sopWords
\context Lyrics = "altos" \lyricsto "altos" \altoWords
\context Lyrics = "tenors" \lyricsto "tenors" \tenorWords
\context Lyrics = "basses" \lyricsto "basses" \bassWords
>>
}

```



## Vocal ensemble template with automatic piano reduction

This template adds an automatic piano reduction to the standard SATB vocal score demonstrated in snippet “Vocal ensemble template”. It demonstrates one of the strengths of LilyPond – you can use a music definition more than once. If any changes are made to the vocal notes (say, `tenorMusic`), then the changes also apply to the piano reduction.

```

\paper {
  top-system-spacing.basic-distance = 10
  score-system-spacing.basic-distance = 20
  system-system-spacing.basic-distance = 20
  last-bottom-spacing.basic-distance = 10
}

global = {
  \key c \major
  \time 4/4
}

```

```

sopMusic = \relative {
  c'4 c c8[( b)] c4
}
sopWords = \lyricmode {
  hi hi hi hi
}

altoMusic = \relative {
  e'4 f d e
}
altoWords = \lyricmode {
  ha ha ha ha
}

tenorMusic = \relative {
  g4 a f g
}
tenorWords = \lyricmode {
  hu hu hu hu
}

bassMusic = \relative {
  c4 c g c
}
bassWords = \lyricmode {
  ho ho ho ho
}

\score {
  <<
    \new ChoirStaff <<
      \new Lyrics = "sopranos" \with {
        % This is needed for lyrics above a staff
        \override VerticalAxisGroup.staff-affinity = #DOWN
      }
      \new Staff = "women" <<
        \new Voice = "sopranos" { \voiceOne << \global \sopMusic >> }
        \new Voice = "altos" { \voiceTwo << \global \altoMusic >> }
      >>
      \new Lyrics = "altos"

      \new Lyrics = "tenors" \with {
        % This is needed for lyrics above a staff
        \override VerticalAxisGroup.staff-affinity = #DOWN
      }
      \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" \sopWords
\context Lyrics = "altos" \lyricsto "altos" \altoWords
\context Lyrics = "tenors" \lyricsto "tenors" \tenorWords
\context Lyrics = "basses" \lyricsto "basses" \bassWords
>>

\new PianoStaff <<
  \new Staff <<
    \set Staff.printPartCombineTexts = ##f
    \partCombine
    << \global \sopMusic >>
    << \global \altoMusic >>
  >>
  \new Staff <<
    \clef bass
    \set Staff.printPartCombineTexts = ##f
    \partCombine
    << \global \tenorMusic >>
    << \global \bassMusic >>
  >>
>>
>>
>>
}

```

hi hi hi hi

ha ha ha ha

hu hu hu hu

ho ho ho ho

## Vocal ensemble template with lyrics aligned below and above the staves

This template is basically the same as the simple “Vocal ensemble template”, with the exception that here all the lyrics lines are placed using `alignAboveContext` and `alignBelowContext`.

```

global = {
  \key c \major
  \time 4/4

```

```

}

sopMusic = \relative c'' {
  c4 c c8[( b)] c4
}
sopWords = \lyricmode {
  hi hi hi hi
}

altoMusic = \relative c' {
  e4 f d e
}
altoWords = \lyricmode {
  ha ha ha ha
}

tenorMusic = \relative c' {
  g4 a f g
}
tenorWords = \lyricmode {
  hu hu hu hu
}

bassMusic = \relative c {
  c4 c g c
}
bassWords = \lyricmode {
  ho ho ho ho
}

\score {
  \new ChoirStaff <<
    \new Staff = "women" <<
      \new Voice = "sopranos" { \voiceOne << \global \sopMusic >> }
      \new Voice = "altos" { \voiceTwo << \global \altoMusic >> }
    >>
    \new Lyrics \with { alignAboveContext = "women" }
      \lyricsto "sopranos" \sopWords
    \new Lyrics \with { alignBelowContext = "women" }
      \lyricsto "altos" \altoWords
    % we could remove the line about this with the line below, since
    % we want the alto lyrics to be below the alto Voice anyway.
    % \new Lyrics \lyricsto "altos" \altoWords

    \new Staff = "men" <<
      \clef bass
      \new Voice = "tenors" { \voiceOne << \global \tenorMusic >> }
      \new Voice = "basses" { \voiceTwo << \global \bassMusic >> }
    >>
    \new Lyrics \with { alignAboveContext = "men" }
      \lyricsto "tenors" \tenorWords
    \new Lyrics \with { alignBelowContext = "men" }

```

```

    \lyricsto "basses" \bassWords
    % again, we could replace the line above this with the line below.
    % \new Lyrics \lyricsto "basses" \bassWords
    >>
}

```



## Vocal ensemble template with verse and refrain

This template creates a score that starts with a solo verse and continues into a refrain for two voices. It also demonstrates the use of spacer rests within the `\global` variable to define meter changes (and other elements common to all parts) throughout the entire score.

```

global = {
  \key g \major

  % verse
  \time 3/4
  s2.*2
  \break

  % refrain
  \time 2/4
  s2*2
  \bar "|"
}

SoloNotes = \relative g' {
  \clef "treble"

  % verse
  g4 g g |
  b4 b b |

  % refrain
  R2*2 |
}

SoloLyrics = \lyricmode {
  One two three |
  four five six |
}

```

```

SopranoNotes = \relative c'' {
  \clef "treble"

  % verse
  R2.*2 |

  % refrain
  c4 c |
  g4 g |
}

SopranoLyrics = \lyricmode {
  la la |
  la la |
}

BassNotes = \relative c {
  \clef "bass"

  % verse
  R2.*2 |

  % refrain
  c4 e |
  d4 d |
}

BassLyrics = \lyricmode {
  dum dum |
  dum dum |
}

\score {
  <<
    \new Voice = "SoloVoice" << \global \SoloNotes >>
    \new Lyrics \lyricsto "SoloVoice" \SoloLyrics

    \new ChoirStaff <<
      \new Voice = "SopranoVoice" << \global \SopranoNotes >>
      \new Lyrics \lyricsto "SopranoVoice" \SopranoLyrics

      \new Voice = "BassVoice" << \global \BassNotes >>
      \new Lyrics \lyricsto "BassVoice" \BassLyrics
    >>
  >>

  \layout {
    ragged-right = ##t
    \context { \Staff
      % these lines prevent empty staves from being printed
      \RemoveEmptyStaves
      \override VerticalAxisGroup.remove-first = ##t
    }
  }
}

```

}  
}  
}

One two three four five six

la la la la

dum dum dum dum

## 10 Keyboard and other multi-staff instruments

See also Section “Keyboard and other multi-staff instruments” in *Notation Reference*.

### Accordion register symbols

Accordion register symbols are available as \markup as well as as standalone music events (as register changes tend to occur between actual music events). Bass registers are not overly standardized. The available commands can be found in ‘Discant symbols’ in the Notation Reference (<https://lilypond.org/doc/v2.24/Documentation/notation/accordion#discant-symbols>).

```
#(use-modules (lily 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" }
    >>
  }
>>
```

The image displays a musical score for a Piano and an Accordion. The Piano part is written on two staves (treble and bass), and the Accordion part is written on a single staff. The score begins with a treble clef and a common time signature (C). The Piano part features a series of chords and notes, including a sequence of eighth and sixteenth notes in the right hand and a bass line in the left hand. The Accordion part uses discant symbols (b, am, gm, a) to indicate specific register changes. The score concludes with a final chord in the Piano part and a final note in the Accordion part.

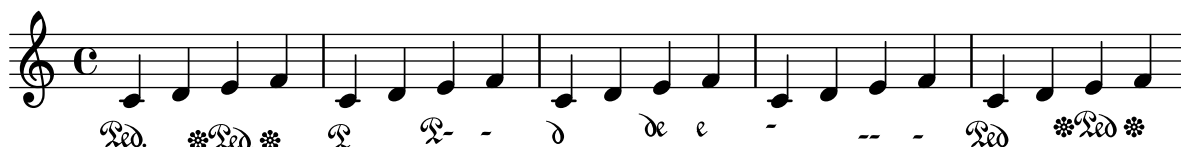
## Changing the text for sustain markings

With the `pedalSustainStrings` context property it is possible to set the text used for pedal down and up. Note that the only valid strings are those found in the list of pedal glyphs – the values shown in this snippet constitute an exhaustive list.

```
sustainNotes = { c4\sustainOn d e\sustainOff\sustainOn f\sustainOff }
```

```
\relative c' {
  \sustainNotes
  \set Staff.pedalSustainStrings = #'("P" "P-" "-")
  \sustainNotes
  \set Staff.pedalSustainStrings = #'("d" "de" "e")
  \sustainNotes
  \set Staff.pedalSustainStrings = #'("M" "M-" "-")
  \sustainNotes
  \set Staff.pedalSustainStrings = #'("Ped" "*Ped" "*")
  \sustainNotes
}

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

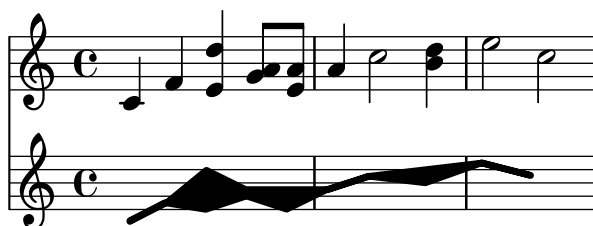


## Clusters

Clusters are a device to denote that a complete range of notes is to be played.

```
fragment = \relative c' {
  c4 f <e d'>4
  <g a>8 <e a> a4 c2 <d b>4
  e2 c
}
```

```
<<
  \new Staff \fragment
  \new Staff \makeClusters \fragment
>>
```



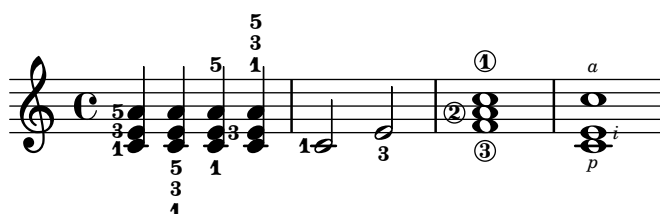
## Controlling the placement of chord fingerings

The placement of fingering numbers can be controlled precisely by using the property `fingeringOrientation`. For fingering orientation to apply, the fingering command must

be used within a chord construct (`<...>`), even for single notes. Orientation for string numbers and right-hand fingerings may be controlled in a similar way by using the properties `stringNumberOrientation` and `strokeFingerOrientation`, respectively.

These properties can be set to a list of one to three values. They control whether fingerings may be placed above (if `up` appears in the list), below (if `down` appears), to the left (if `left` appears), or to the right (if `right` appears). Conversely, if a location is not listed, no fingering is placed there. LilyPond takes these constraints and works out the best placement for the fingering of the notes of the following chords. Note that `left` and `right` are mutually exclusive – fingerings may be placed only on one side or the other, not both.

```
\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
  \set stringNumberOrientations = #'(up left down)
  <f\3 a\2 c\1>1
  \set strokeFingerOrientations = #'(down right up)
  <c\rightHandFinger 1 e\rightHandFinger 2 c'\rightHandFinger 4 >
}
```



## Creating slurs across voices

In some situations it is necessary to create slurs between notes from different voices. The solution is to add invisible notes to one of the voices, using `\hideNotes`.

This example is measure 235 of the Ciaccona from Bach's second partita for solo violin, BWV 1004.

```
\relative c' {
  <<
  {
    d16( a') s a s a[ s a] s a[ s a]
  }
  \\\
  {
    \slurUp
    bes,16[ s e](
    \hideNotes a)
    \unHideNotes f[(

```



```

\hideNotes a)
\unHideNotes fis](
\hideNotes a)
\unHideNotes g[(
\hideNotes a)
\unHideNotes gis](
\hideNotes a)
}
>>
}

```



## Cross-staff chords – beaming problems workaround

Sometimes it is better to use stems from the ‘other’ staff for creating cross-staff chords to trick LilyPond’s beam collision detector. In the following snippet, if the stems from the lower staff were used instead, it would be necessary to explicitly use

```
\override Staff.Beam.collision-voice-only = ##t
```

so that LilyPond doesn’t move the beams.

```

\new PianoStaff <<
\new Staff = up \relative c' <<
{ r4
  \override Stem.cross-staff = ##t
  \override Stem.length = #19 % this is in half-spaces,
    % so it makes stems 9.5 staffspaces long
  \override Stem.Y-offset = #-6 % stems are normally lengthened
    % upwards, so here we must lower the stem by the amount
    % equal to the lengthening - in this case (19 - 7) / 2
    % (7 is default stem length)
  e e e }
{ s4
  \change Staff = "bottom"
  \override NoteColumn.ignore-collision = ##t
  c, c c
}
>>

\new Staff = bottom \relative c' {
  \clef bass
  \voiceOne
  g8 a g a g a g a
}
>>

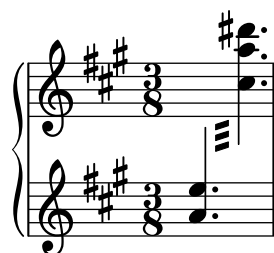
```



## 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
      }
    }
  }
>>
```



## Fine-tuning pedal brackets

The appearance of pedal brackets may be altered in different ways.

```
\paper {
  ragged-right = ##f
}

\relative c'' {
  c2\sostenutoOn c
  c2\sostenutoOff c
  c2\tweak shorten-pair #'(-7 . -2) \sostenutoOn c
  c2\sostenutoOff c
```

```

c2\tweak edge-height #'(0 . 3) \sostenutoOn c
c2\sostenutoOff c
}

```



## 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.

The following example typesets measure 65 of Debussy's prelude *Les collines d'Anacapri*.

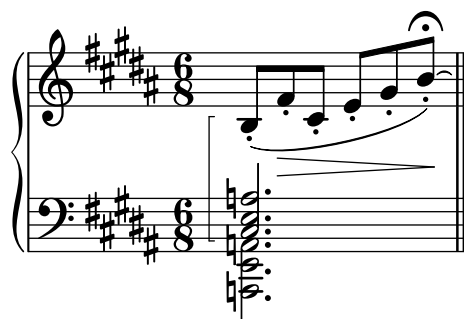
```

\new PianoStaff <<
  \set PianoStaff.connectArpeggios = ##t
  \override PianoStaff.Arpeggio.stencil =
    #ly:arpeggio::brew-chord-bracket

  \new Staff \relative c' {
    \key b \major
    \time 6/8
    b8-.(\arpeggio fis'-.\> cis-.
      e-. gis-. b-.)\!\fermata^\laissezVibrer \bar "||"
  }

  \new Staff \relative c' {
    \clef bass
    \key b \major
    << { <a e cis>2.\arpeggio } \
      { <a, e a,>2. } >>
  }
>>

```



## Jazz combo template

This is quite an advanced template, for a jazz ensemble. Note that all instruments use `\key c \major`. This refers to the key in concert pitch; the key will be automatically transposed if the music is within a `\transpose` section.

```

\header {
  title = "Song"
}

```

```

subtitle = "(tune)"
composer = "Me"
meter = "moderato"
piece = "Swing"
tagline = \markup \column {
    "LilyPond example file by Amelie Zapf,"
    "Berlin 07/07/2003" }
}

% To make the example display properly in the documentation.
\paper {
    paper-width = 130\mm
    paper-height = 205\mm
}

% #(set-global-staff-size 16)

\include "english.ly"

%%%%%%%%%%%%%% Some macros %%%%%%%%%%%%%%%

sl = { \override NoteHead.style = #'slash
       \hide Stem }
nsl = { \revert NoteHead.style
        \undo \hide Stem }
crOn = \override NoteHead.style = #'cross
crOff = \revert NoteHead.style

% Insert chord name style stuff here.

jazzChords = { }

%%%%%%%%%%%%%% Keys 'n' thangs %%%%%%%%%%%%%%%

global = { \time 4/4 }

Key = { \key c \major }

% ##### Horns #####

% ----- Trumpet -----
trpt = \transpose c d \relative c' {
    \Key
    c1 | c | c |
}
trpHarmony = \transpose c' d {
    \jazzChords
}
trumpet = {
    \global

```

```

\clef treble
\trpt
}

% ----- Alto Saxophone -----
alto = \transpose c a \relative c' {
  \Key
  c1 | c | c |
}
altoHarmony = \transpose c' a {
  \jazzChords
}
altoSax = {
  \global
  \clef treble
  \alto
}

% ----- Baritone Saxophone -----
bari = \transpose c a' \relative c {
  \Key
  c1 | c1 |
  \sl d4^"Solo" d d d \ns1 |
}
bariHarmony = \transpose c' a \chordmode {
  \jazzChords
  s1 | s |
  d2:maj e:m7 |
}
bariSax = {
  \global
  \clef treble
  \bari
}

% ----- Trombone -----
tbone = \relative c {
  \Key
  c1 | c | c |
}
tboneHarmony = \chordmode {
  \jazzChords
}
trombone = {
  \global
  \clef bass
  \tbone
}

% ##### Rhythm Section #####

% ----- Guitar -----

```

```

gtr = \relative c'' {
  \Key
  c1 |
  \sl b4 b b b \ns1 |
  c1 |
}
gtrHarmony = \chordmode {
  \jazzChords
  s1 | c2:min7+ d2:maj9 | s1 |
}
guitar = {
  \global
  \clef treble
  \gtr
}

%% ----- Piano -----
rhUpper = \relative c'' {
  \voiceOne
  \Key
  c1 | c | c |
}
rhLower = \relative c' {
  \voiceTwo
  \Key
  e1 | e | e |
}

lhUpper = \relative c' {
  \voiceOne
  \Key
  g1 | g | g |
}
lhLower = \relative c {
  \voiceTwo
  \Key
  c1 | c | c |
}

PianoRH = {
  \clef treble
  \global
  <<
    \new Voice = "one" \rhUpper
    \new Voice = "two" \rhLower
  >>
}
PianoLH = {
  \clef bass
  \global
  <<
    \new Voice = "one" \lhUpper

```

```

    \new Voice = "two" \lhLower
  >>
}

piano = <<
  \new Staff = "upper" \PianoRH
  \new Staff = "lower" \PianoLH
>>

% ----- Bass Guitar -----
Bass = \relative c {
  \Key
  c1 | c | c |
}
bass = {
  \global
  \clef bass
  \Bass
}

% ----- Drums -----
up = \drummode {
  \voiceOne
  hh4 <hh sn> hh <hh sn> |
  hh4 <hh sn> hh <hh sn> |
  hh4 <hh sn> hh <hh sn> |
}
down = \drummode {
  \voiceTwo
  bd4 s bd s |
  bd4 s bd s |
  bd4 s bd s |
}

drumContents = {
  \global
  <<
    \new DrumVoice \up
    \new DrumVoice \down
  >>
}

%%%%%%%%%% It All Goes Together Here %%%%%%%%%%%

\book { % For the LilyPond documentation.
  \score {
    <<
      \new StaffGroup = "horns" <<
        \new Staff = "trumpet" \with { instrumentName = "Trumpet" }
        \trumpet
        \new Staff = "altosax" \with { instrumentName = "Alto Sax" }

```

```

    \altoSax
    \new ChordNames = "barichords" \with { instrumentName = "Bari Sax" }
    \bariHarmony
    \new Staff = "barisax" \with { instrumentName = "Bari Sax" }
    \bariSax
    \new Staff = "trombone" \with { instrumentName = "Trombone" }
    \trombone
  >>

  \new StaffGroup = "rhythm" <<
    \new ChordNames = "chords" \with { instrumentName = "Guitar" }
    \gtrHarmony
    \new Staff = "guitar" \with { instrumentName = "Guitar" }
    \guitar
    \new PianoStaff = "piano" \with {
      instrumentName = "Piano"
      midiInstrument = "acoustic grand"
    } \piano
    \new Staff = "bass" \with { instrumentName = "Bass" }
    \bass
    \new DrumStaff \with { instrumentName = "Drums" }
    \drumContents
  >>
>>

\layout {
  \context {
    \Staff
    \RemoveEmptyStaves
  }
  \context {
    \Score
    \override BarNumber.padding = 3
    \override RehearsalMark.padding = 2
    skipBars = ##t
  }
}
\midi { }
}

```



## Song

(tune)

Me

moderato  
Swing

Trumpet

Alto Sax

Bari Sax

Trombone

Guitar

Piano

Bass

Drums

B<sup>Δ</sup> C#m<sup>7</sup>

Cm<sup>Δ</sup> D<sup>Δ</sup><sup>9</sup>

LilyPond example file by Amelie Zapf,  
Berlin 07/07/2003

## Laissez vibrer ties

Laissez vibrer ties have a fixed size. Their positioning can be tuned using the tie-configuration property.

See also snippet “Longer laissez vibrer ties”.

```
\relative c' {
  <c e g>4\laissezVibrer r <c f g>\laissezVibrer r
  <c d f g>4\laissezVibrer r <c d f g>4.\laissezVibrer r8

  <c d e f>4\laissezVibrer r
  \override LaissezVibrerTieColumn.tie-configuration
    = #`((-7 . ,DOWN)
```

```

        (-5 . ,DOWN)
        (-3 . ,UP)
        (-1 . ,UP))
    <c d e f>4\laissezVibrer r
}

```



## Piano template (simple)

Here is a simple piano staff with some notes.

```

upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

```

```

lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  a2 c
}

```

```

\score {
  \new PianoStaff \with { instrumentName = "Piano" }
  <<
    \new Staff = "upper" \upper
    \new Staff = "lower" \lower
  >>
  \layout { }
  \midi { }
}

```



## Piano template with centered lyrics

Instead of having a full staff for the melody and lyrics, lyrics can be centered between the staves of a piano staff.

```

upper = \relative c'' {
  \clef treble
  \key c \major

```

```

\time 4/4

a4 b c d
}

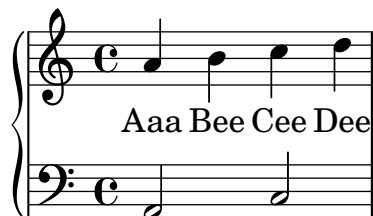
lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  a2 c
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

\score {
  \new PianoStaff <<
    \new Staff = upper { \new Voice = "singer" \upper }
    \new Lyrics \lyricsto "singer" \text
    \new Staff = lower { \lower }
  >>
  \layout { }
  \midi { }
}

```



## Piano template with melody and lyrics

Here is a typical song format: one staff with the melody and lyrics, with piano accompaniment underneath.

```

melody = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

upper = \relative c'' {
  \clef treble

```

```

\key c \major
\time 4/4

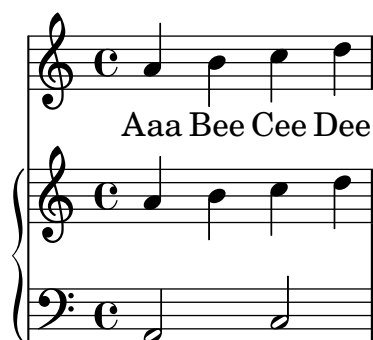
a4 b c d
}

lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  a2 c
}

\score {
  <<
    \new Voice = "mel" { \autoBeamOff \melody }
    \new Lyrics \lyricsto mel \text
    \new PianoStaff <<
      \new Staff = "upper" \upper
      \new Staff = "lower" \lower
    >>
  >>
  \layout {
    \context { \Staff \RemoveEmptyStaves }
  }
  \midi { }
}

```



## Removing brace on first line of piano score

This snippet removes the first brace from a PianoStaff or a GrandStaff, together with the clefs. It may be useful when cutting and pasting the engraved image into existing music.

The code uses `\alterBroken` to hide the brace delimiter at the beginning.

```

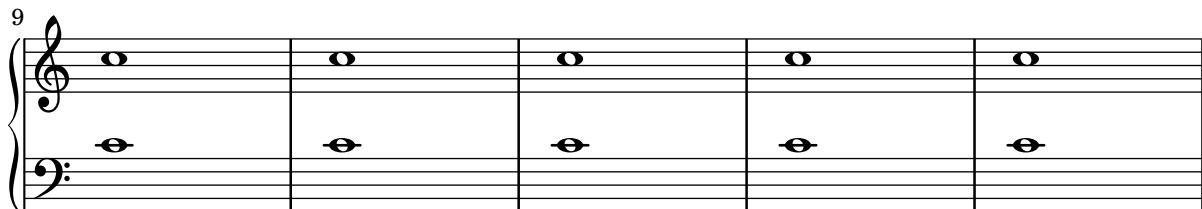
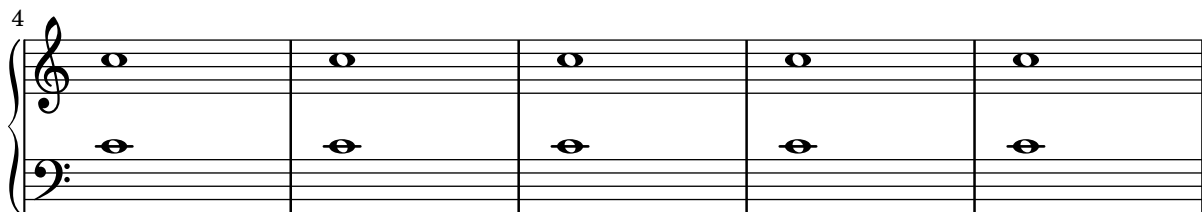
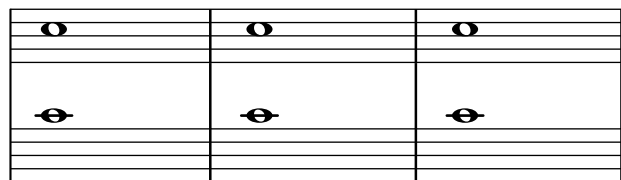
someMusic = {
  \once \omit Staff.Clef
  \once \omit Staff.TimeSignature
  \repeat unfold 3 c1 \break
  \repeat unfold 5 c1 \break
  \repeat unfold 5 c1
}

```

```

\score {
  \new PianoStaff
  <<
    \new Staff = "right" \relative c'' \someMusic
    \new Staff = "left" \relative c' { \clef F \someMusic }
  >>
  \layout {
    indent=75\mm
    \context {
      \PianoStaff
      \alterBroken transparent #'(#t) SystemStartBrace
    }
  }
}

```



## Using \autoChange with more than one voice

Here is a demonstration of how to use \autoChange with more than one voice.

```

\score {
  \new PianoStaff
  <<
    \new Staff = "up" {
      <<
        \set Timing.beamExceptions = #'()
        \set Timing.beatStructure = #'(4)
        \new Voice {
          \voiceOne
          \autoChange
          \relative c' {
            g8 a b c d e f g
            g,8 a b c d e f g
          }
        }
      >>
    }
  >>
}

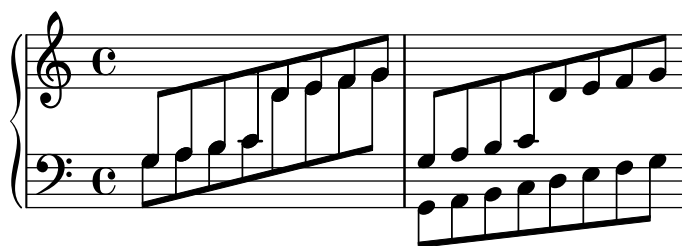
```

```

\new Voice {
  \voiceTwo
  \autoChange
  \relative c' {
    g8 a b c d e f g
    g,,8 a b c d e f g
  }
}
>>
}

\new Staff = "down" {
  \clef bass
}
>>
}

```



## Vocal ensemble template with automatic piano reduction

This template adds an automatic piano reduction to the standard SATB vocal score demonstrated in snippet “Vocal ensemble template”. It demonstrates one of the strengths of LilyPond – you can use a music definition more than once. If any changes are made to the vocal notes (say, `tenorMusic`), then the changes also apply to the piano reduction.

```

\paper {
  top-system-spacing.basic-distance = 10
  score-system-spacing.basic-distance = 20
  system-system-spacing.basic-distance = 20
  last-bottom-spacing.basic-distance = 10
}

global = {
  \key c \major
  \time 4/4
}

sopMusic = \relative {
  c''4 c c8[( b)] c4
}
sopWords = \lyricmode {
  hi hi hi hi
}

altoMusic = \relative {

```

```

    e'4 f d e
}
altoWords = \lyricmode {
    ha ha ha ha
}

tenorMusic = \relative {
    g4 a f g
}
tenorWords = \lyricmode {
    hu hu hu hu
}

bassMusic = \relative {
    c4 c g c
}
bassWords = \lyricmode {
    ho ho ho ho
}

\score {
  <<
    \new ChoirStaff <<
      \new Lyrics = "sopranos" \with {
        % This is needed for lyrics above a staff
        \override VerticalAxisGroup.staff-affinity = #DOWN
      }
      \new Staff = "women" <<
        \new Voice = "sopranos" { \voiceOne << \global \sopMusic >> }
        \new Voice = "altos" { \voiceTwo << \global \altoMusic >> }
      >>
      \new Lyrics = "altos"

      \new Lyrics = "tenors" \with {
        % This is needed for lyrics above a staff
        \override VerticalAxisGroup.staff-affinity = #DOWN
      }
      \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" \sopWords
      \context Lyrics = "altos" \lyricsto "altos" \altoWords
      \context Lyrics = "tenors" \lyricsto "tenors" \tenorWords
      \context Lyrics = "basses" \lyricsto "basses" \bassWords
    >>

    \new PianoStaff <<
      \new Staff <<

```

```

\set Staff.printPartCombineTexts = ##f
\partCombine
<< \global \sopMusic >>
<< \global \altoMusic >>
>>
\new Staff <<
  \clef bass
  \set Staff.printPartCombineTexts = ##f
  \partCombine
  << \global \tenorMusic >>
  << \global \bassMusic >>
>>
>>
>>
}

```

The image shows a musical score for a multi-staff instrument, likely a keyboard or a multi-staff vocal instrument. The score is written in common time (C) and consists of four staves. The first two staves are grouped by a brace on the left, and the last two staves are also grouped by a brace on the left. The first staff has a treble clef, and the second staff has a bass clef. The third staff has a treble clef, and the fourth staff has a bass clef. The lyrics are: "hi hi hi hi" on the first staff, "ha ha ha ha" on the second staff, "hu hu hu hu" on the third staff, and "ho ho ho ho" on the fourth staff. The notes are: first staff (quarter, quarter, quarter, quarter), second staff (quarter, quarter, quarter, quarter), third staff (quarter, quarter, quarter, quarter), and fourth staff (quarter, quarter, quarter, quarter). The notes are: first staff (G4, A4, B4, C5), second staff (G3, A3, B3, C4), third staff (G3, A3, B3, C4), and fourth staff (G3, A3, B3, C4).



## 11 Unfretted string instruments

See also Section “Unfretted string instruments” in *Notation Reference*.

### Creating slurs across voices

In some situations it is necessary to create slurs between notes from different voices. The solution is to add invisible notes to one of the voices, using `\hideNotes`.

This example is measure 235 of the Ciaccona from Bach’s second partita for solo violin, BWV 1004.

```
\relative c' {
  <<
    {
      d16( a') s a s a[ s a] s a[ s a]
    }
    \\\
    {
      \slurUp
      bes,16[ s e](
      \hideNotes a)
      \unHideNotes f[(
      \hideNotes a)
      \unHideNotes fis](
      \hideNotes a)
      \unHideNotes g[(
      \hideNotes a)
      \unHideNotes gis](
      \hideNotes a)
    }
  >>
}
```



### Dotted harmonics

Artificial harmonics using `\harmonic` do not show dots. To override this behavior, set the context property `harmonicDots`.

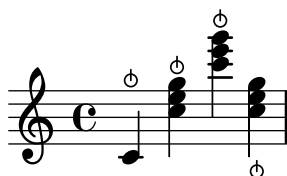
```
\relative c' '' {
  \time 3/4
  \key f \major
  \set harmonicDots = ##t
  <bes f'\harmonic>2. ~
  <bes f'\harmonic>4. <a e'\harmonic>8( <gis dis'\harmonic> <g d'\harmonic>)
  <fis cis'\harmonic>2.
  <bes f'\harmonic>2.
}
```



## Snap pizzicato (“Bartok” pizzicato)

A snap pizzicato (also known as “Bartok pizzicato”) is a “strong pizzicato where the string is plucked vertically by snapping and rebounds off the fingerboard of the instrument” (Wikipedia). It is denoted by a circle with a vertical line going from the center upwards outside the circle.

```
\relative c' {
  c4\snappizzicato
  <c' e g>4\snappizzicato
  <c' e g>4^\snappizzicato
  <c, e g>4_\snappizzicato
}
```



## String quartet template (simple)

This template demonstrates a simple string quartet. It also uses a `\global` section for time and key signatures.

See also snippet “String quartet template with separate parts”.

```
global= {
  \time 4/4
  \key c \major
}

violinOne = \new Voice \relative c'' {
  c2 d
  e1
  \bar "|."
}

violinTwo = \new Voice \relative c'' {
  g2 f
  e1
  \bar "|."
}

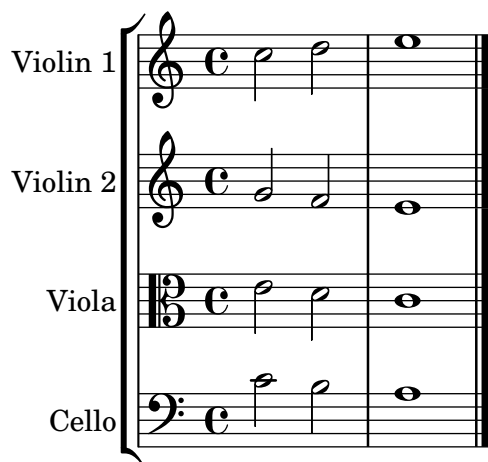
viola = \new Voice \relative c' {
  \clef alto
  e2 d
  c1
  \bar "|."
}
```

```

cello = \new Voice \relative c' {
  \clef bass
  c2 b
  a1
  \bar "|."
}

\score {
  \new StaffGroup <<
    \new Staff \with { instrumentName = "Violin 1" }
    << \global \violinOne >>
    \new Staff \with { instrumentName = "Violin 2" }
    << \global \violinTwo >>
    \new Staff \with { instrumentName = "Viola" }
    << \global \viola >>
    \new Staff \with { instrumentName = "Cello" }
    << \global \cello >>
  >>
  \layout { }
  \midi { }
}

```



## String quartet template with separate parts

The “String quartet template (simple)” snippet produces a nice string quartet, but what if you need to print parts? This new template demonstrates how to use the `\tag` feature to easily split a piece into individual parts.

For technical reasons, multiple output files cannot be shown here for a single snippet, which means that the template below unifies the code for separate files. The file names are contained in comments at the beginning of each file.

`piece.ly` contains all the music definitions. The other files – `score.ly`, `vn1.ly`, `vn2.ly`, `vla.ly`, and `vlc.ly` – produce the full score and the four parts.

Do not forget to remove specified comments when using separate files!

```

% piece.ly
% (This is the global definitions file.)

```

```

global= {

```

```

\time 4/4
\key c \major
}

Violinone = \new Voice \relative c' {
  c2 d e1
  \bar "|."
}

Violintwo = \new Voice \relative c' {
  g2 g e1
  \bar "|."
}

Viola = \new Voice \relative c' {
  \clef alto
  e2 d c1
  \bar "|."
}

Cello = \new Voice \relative c' {
  \clef bass
  c2 b a1
  \bar "|."
}

music = <<
  \tag #'score \tag #'vn1
  \new Staff \with { instrumentName = "Violin 1" }
    << \global \Violinone >>

  \tag #'score \tag #'vn2
  \new Staff \with { instrumentName = "Violin 2" }
    << \global \Violintwo >>

  \tag #'score \tag #'vla
  \new Staff \with { instrumentName = "Viola" }
    << \global \Viola >>

  \tag #'score \tag #'vlc
  \new Staff \with { instrumentName = "Cello" }
    << \global \Cello >>
>>

% These are the other files you need to save on your computer

% score.ly
% (This is the main file.)

% Uncomment the line below when using a separate file.
% \include "piece.ly"

```

```

#(set-global-staff-size 14)

\score {
  \new StaffGroup \keepWithTag #'score \music
  \layout { }
  \midi { }
}

%{ Uncomment this block when using separate files.

% vn1.ly
% (This is the Violin 1 part file.)

\include "piece.ly"
\score {
  \keepWithTag #'vn1 \music
  \layout { }
}

% vn2.ly
% (This is the Violin 2 part file.)

\include "piece.ly"
\score {
  \keepWithTag #'vn2 \music
  \layout { }
}

% vla.ly
% (This is the Viola part file.)

\include "piece.ly"
\score {
  \keepWithTag #'vla \music
  \layout { }
}

% vlc.ly
% (This is the Cello part file.)

\include "piece.ly"
\score {
  \keepWithTag #'vlc \music
  \layout { }
}

%}

```

Violin 1

Violin 2

Viola

Cello

The image shows a musical score for four string instruments: Violin 1, Violin 2, Viola, and Cello. The score is written in common time (C) and consists of two measures. Violin 1 plays a half note G4, followed by a half note A4, and then a whole note B4. Violin 2 plays a half note E4, followed by a half note F4, and then a whole note G4. Viola plays a half note C3, followed by a half note D3, and then a whole note E3. Cello plays a half note G2, followed by a half note A2, and then a whole note B2. The score is written on four staves, with Violin 1 and Violin 2 in treble clef, and Viola and Cello in bass clef. The key signature is one flat (Bb).

## 12 Fretted string instruments

See also Section “Fretted string instruments” in *Notation Reference*.

### Adding fingerings to a score

Fingering instructions can be entered using a simple syntax.

```
\relative c' ' {
  c4-1 d-2 f-4 e-3
}
```

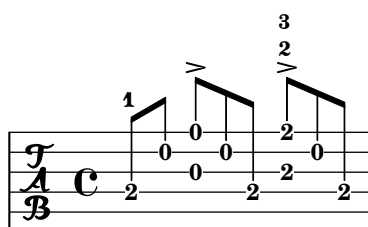


### Adding fingerings to tablatures

To add fingerings to tablatures, use a combination of `\markup` and `\finger`.

```
one = \markup { \finger 1 }
two = \markup { \finger 2 }
threeTwo = \markup {
  \override #'(baseline-skip . 2)
  \column {
    \finger 3
    \finger 2
  }
}

\score {
  \new TabStaff {
    \tabFullNotation
    \stemUp
    e8\4^\one b\2 <g\3 e'\1>^\two[ b\2 e\4]
    <a\3 fis'\1>^\threeTwo[ b\2 e\4]
  }
}
```



### Adding markups in a tablature

By default, markups are not displayed in a tablature.

To make them appear, revert the stencil property of the TextScript grob in the TabStaff context.

```
high = { r4 r8 <g c'> q r8 r4 }
low = { c4 r4 c8 r8 g,8 b, }
pulse = { s8^"1" s^"&" s^"2" s^"&" s^"3" s^"&" s^"4" s^"&" }
```



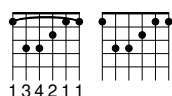


## Automatic fretboards barré

When automatic fretboards are used, barré indicators are drawn whenever one finger is responsible for multiple strings.

If no finger indications are given in the chord from which the automatic fretboard is created, no barré indicators are included, because there is no way to identify where barrés should be placed.

```
\new FretBoards {
  <f,-1 c-3 f-4 a-2 c'-1 f'-1>1
  <f, c f a c' f'>1
}
```

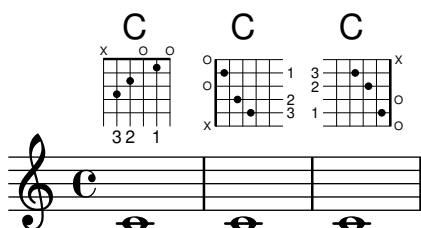


## Changing fret orientations

Fret diagrams can be oriented in three ways. By default the top string or fret in the different orientations will be aligned.

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

```
<<
\chords {
  c1
  c1
  c1
}
\new FretBoards \chordmode {
  c1
  \override FretBoard.fret-diagram-details.orientation =
    #'landscape
  c1
  \override FretBoard.fret-diagram-details.orientation =
    #'opposing-landscape
  c1
}
\new Voice {
  c'1
  c'1
  c'
}
>>
```



## Chord changes 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 }
>>
```

The image displays three staves of musical notation, each representing a different chord change scenario for a C major chord. Each staff begins with a treble clef and a common time signature 'C'. Above each staff is a fretboard diagram for a C major chord, showing fingerings 3, 2, 1 on the strings. The first staff shows the chord at the beginning. The second staff shows the chord at the beginning and middle. The third staff shows the chord at the beginning and middle.

## Chord glissando in tablature

Slides for chords are indicated by default in both Staff and TabStaff.

String numbers may be necessary for TabStaff because automatic string calculations are different for chords and for single notes.

```
myMusic = \relative c' {
  <c e g>1 \glissando <f a c>
  <cis, eis gis>1 \glissando <f a c>
  <cis eis gis>1 \glissando <f a c\3>
}
```

```
\score {
  <<
```

```

\new Staff {
  \clef "treble_8"
  \omit StringNumber
  \myMusic
}
\new TabStaff \myMusic
>>
}

\score {
  <<
    \new Staff {
      \clef "treble_8"
      \omit StringNumber
      \myMusic
    }
    \new TabStaff \with { \override Glissando.style = #'none } {
      \myMusic
    }
  >>
}

```

## Chords with stretched fingering for FretBoards and TabVoice

Sometimes chords with a stretched fingering are required. If not otherwise specified the context property `maximumFretStretch` is set to value 4, though, resulting in a warning about “No string for pitch ...”, and the note is omitted. You may set `maximumFretStretch` to an appropriate value or explicitly assign string numbers to all notes of a chord to fix that.

```

% The code below prints two warnings for the second chord,
% which may be omitted by uncommenting the following line.
%
% #(for-each (lambda (x) (ly:expect-warning "No string for pitch")) (iota 2))

```

```

mus = {
  <c' bes'>
  <c'\2 bes'>
  \set maximumFretStretch = 5
}

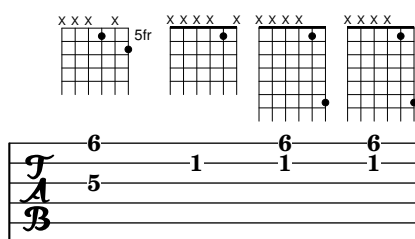
```

```

<c' bes'>
<c'\2 bes'\1>
}

<<
  \new FretBoards \mus
  \new TabVoice \mus
>>

```



## Controlling the placement of chord fingerings

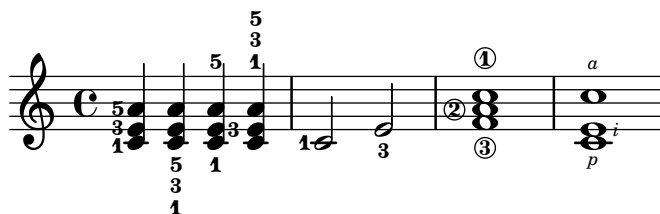
The placement of fingering numbers can be controlled precisely by using the property `fingeringOrientation`. For fingering orientation to apply, the fingering command must be used within a chord construct (`<...>`), even for single notes. Orientation for string numbers and right-hand fingerings may be controlled in a similar way by using the properties `stringNumberOrientation` and `strokeFingerOrientation`, respectively.

These properties can be set to a list of one to three values. They control whether fingerings may be placed above (if `up` appears in the list), below (if `down` appears), to the left (if `left` appears), or to the right (if `right` appears). Conversely, if a location is not listed, no fingering is placed there. LilyPond takes these constraints and works out the best placement for the fingering of the notes of the following chords. Note that `left` and `right` are mutually exclusive – fingerings may be placed only on one side or the other, not both.

```

\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
  \set stringNumberOrientations = #'(up left down)
  <f\3 a\2 c\1>1
  \set strokeFingerOrientations = #'(down right up)
  <c\rightHandFinger 1 e\rightHandFinger 2 c'\rightHandFinger 4 >
}

```



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, and therefore can be omitted in property overrides.

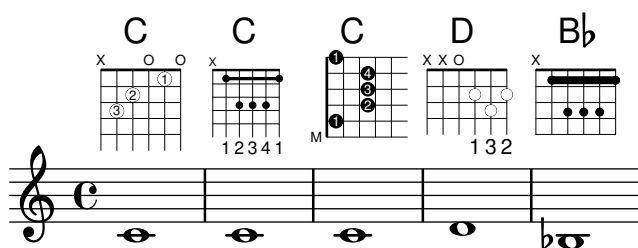
```
\new ChordNames {
  \chordmode { c1 | c | c | d | bes }
}

\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
    \oo FretBoard.size #1.0
    \oo FretBoard.fret-diagram-details.barre-type #'straight
    \oo FretBoard.fret-diagram-details.dot-color #'black
    \oo FretBoard.fret-diagram-details.finger-code #'below-string
    c'
    \oo FretBoard.fret-diagram-details.barre-type #'none
    \oo FretBoard.fret-diagram-details.number-type #'arabic
    \oo FretBoard.fret-diagram-details.orientation #'landscape
    \oo FretBoard.fret-diagram-details.mute-string "M"
    \oo FretBoard.fret-diagram-details.label-dir #LEFT
    \oo FretBoard.fret-diagram-details.dot-color #'black
    c'
    \oo FretBoard.fret-diagram-details.finger-code #'below-string
    \oo FretBoard.fret-diagram-details.dot-radius #0.35
    \oo FretBoard.fret-diagram-details.dot-position #0.5
    \oo FretBoard.fret-diagram-details.fret-count #3
  }
}
```

```

d
  \oo FretBoard.fret-diagram-details.barre-type #'straight
  \oo FretBoard.fret-diagram-details.finger-code #'none
  \oo FretBoard.fret-diagram-details.dot-radius #0.25
  \oo FretBoard.fret-diagram-details.dot-color #'black
  \oo FretBoard.fret-diagram-details.string-overhang #0.
  \oo FretBoard.fret-diagram-details.barre-thickness #2.
bes
}
}
\new Voice {
  c'1 | c' | c' | d' | bes
}
>>

```



## Customizing markup fret diagrams

Fret diagram properties can be modified by setting the `fret-diagram-details` property. For markup fret diagrams, overrides can be applied to the `Voice.TextScript` object or directly to the markup.

```

<<
\chords { c1 | c | c | d }

\new Voice = "mel" {
  \textLengthOn
  % Set global properties of fret diagram
  \override TextScript.size = 1.2
  \override TextScript.fret-diagram-details.finger-code = #'in-dot
  \override TextScript.fret-diagram-details.dot-color = #'white

  %% C major for guitar, no barre, using defaults
  % terse style
  c'1^\markup { \fret-diagram-terse "x;3-3;2-2;o;1-1;o;" }

  %% C major for guitar, barred on third fret
  % verbose style
  % size 1.0
  % roman fret label, finger labels below string, straight barre
  c'1^\markup {
    % standard size
    \override #'(size . 1.0) {
      \override #'(fret-diagram-details . (
        (number-type . roman-lower)
        (finger-code . in-dot)

```

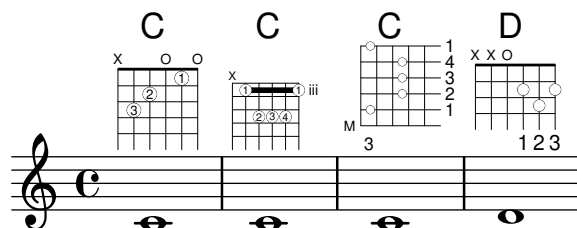
```

        (barre-type . straight))) {
  \fret-diagram-verbose #'((mute 6)
    (place-fret 5 3 1)
    (place-fret 4 5 2)
    (place-fret 3 5 3)
    (place-fret 2 5 4)
    (place-fret 1 3 1)
    (barre 5 1 3))
  }
}
}

%% C major for guitar, barred on third fret
% verbose style
% landscape orientation, arabic numbers, M for mute string
% no barre, fret label down or left, small mute label font
c'1^\markup {
  \override #'(fret-diagram-details . (
    (finger-code . below-string)
    (number-type . arabic)
    (label-dir . -1)
    (mute-string . "M")
    (orientation . landscape)
    (barre-type . none)
    (xo-font-magnification . 0.4)
    (xo-padding . 0.3))) {
    \fret-diagram-verbose #'((mute 6)
      (place-fret 5 3 1)
      (place-fret 4 5 2)
      (place-fret 3 5 3)
      (place-fret 2 5 4)
      (place-fret 1 3 1)
      (barre 5 1 3))
    }
  }
}

%% simple D chord
% terse style
% larger dots, centered dots, fewer frets
% label below string
d'1^\markup {
  \override #'(fret-diagram-details . (
    (finger-code . below-string)
    (dot-radius . 0.35)
    (dot-position . 0.5)
    (fret-count . 3))) {
    \fret-diagram-terse "x;x;o;2-1;3-2;2-3;"
  }
}
}
>>

```

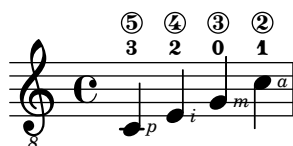


## Fingerings, string indications, and right-hand fingerings

This example combines left-hand fingering, string indications, and right-hand fingering.

```
#(define RH rightHandFinger)
```

```
\relative c {
  \clef "treble_8"
  <c-3\5\RH 1 >4
  <e-2\4\RH 2 >4
  <g-0\3\RH 3 >4
  <c-1\2\RH 4 >4
}
```



## Flamenco notation

For flamenco guitar, some special notation is used.

- A *golpe* symbol indicates a slap on the guitar body with the nail of the ring finger.
- An arrow indicates (the direction of) strokes.
- Different letters for fingering are used (“p”: thumb, “i”: index finger, “m”: middle finger, “a”: ring finger and “x”: little finger).
- Marking 3- and 4-finger *rasgueados*: stroke upwards with all fingers, ending with an up-and down using the index finger.
- *Abanicos* are strokes (in tuples) with thumb (down), little and index finger (both up). There’s also an *abanico 2* where middle and ring finger are used instead of the little finger.
- *Alza pua* indicates fast playing with the thumb.

Most figures use arrows in combination with fingering; with abanicos and rasgueados, note heads are printed only for the first chord.

This snippet contains some header-like code that can be copied as `flamenco.ly` and included in source files.

```
%%%%%%%% Cut here ----- Start of `flamenco.ly`.
```

```
% Text indicators.
```

```
abanico = ^\markup \small { \italic Abanico }
```

```
rasgueado = ^\markup \small { \italic Ras. }
```

```
alzapua = ^\markup \small { \italic Alzapua }
```

```
% Finger stroke symbols.
```

```
strokeUp = \markup {
```



```

\combine
  \override #'(thickness . 1.3) \draw-line #'(0 . 2)
  \raise #2 \arrow-head #Y #UP ##f }
strokeDown = \markup {
  \combine
    \arrow-head #Y #DOWN ##f
    \override #'(thickness . 1.3) \draw-line #'(0 . 2) }

% Golpe symbol.
golpe = \markup {
  \filled-box #'(0 . 1) #'(0 . 1) #0
  \hspace #-1.6
  \with-color #white
  \filled-box #'(0.15 . 0.85) #'(0.15 . 0.85) #0
}

% Strokes, fingers, and golpe command.
RHp = \rightHandFinger #1
RHi = \rightHandFinger #2
RHm = \rightHandFinger #3
RHa = \rightHandFinger #4
RHx = \rightHandFinger #5
RHu = \rightHandFinger \strokeUp
RHd = \rightHandFinger \strokeDown
RHg = \rightHandFinger \golpe

% Various shorthands.
tupletOff = {
  \once \omit TupletNumber
  \once \omit TupletBracket
}

tupletsOff = {
  \omit TupletNumber
  \override TupletBracket.bracket-visibility = #'if-no-beam
}

tupletsOn = {
  \override TupletBracket.bracket-visibility = #'default
  \undo \omit TupletNumber
}

headsOff = {
  \hide TabNoteHead
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}

headsOn = {
  \override TabNoteHead.transparent = ##f
  \override NoteHead.transparent = ##f
  \override NoteHead.no-ledgers = ##f
}

```

```
}
```

```
%%%%%%%% Cut here ----- End of `flamenco.ly`.
```

```
part = \relative c' {
  \set strokeFingerOrientations = #'(up)
  \key a\major

  <a, e' a cis e\RHu\RH>8
    <a e' a cis e\RHd\RH>8
    r4
    r2^\markup\golpe |
  <a e' a cis e\RHu\RH>8
    <a e' a cis e\RHd\RH>8
    <a e' a cis e\RHu\RH\RHg>8
    <a e' a cis e\RHd\RH>8
    r2 |
  <a e' a cis e\RHu\RH>16\rasgueado
    \headsOff
    <a e' a cis e\RHu\RH>
    <a e' a cis e\RHu\RH>
    <a e' a cis e\RHd\RH>~
    \headsOn
    <a e' a cis e>2
    r4 |
  \tupletOff
  \tuplet 5/4 {
    <a e' a cis e\RHu\RH>16\rasgueado
    \headsOff
    <a e' a cis e\RHu\RH>
    <a e' a cis e\RHu\RH>
    <a e' a cis e\RHu\RH>
    <a e' a cis e\RHd\RH>~
    \headsOn
  }
  <a e' a cis e>2
  r4 |
  <>\abánico
  \tupletsOff
  \repeat unfold 4 {
    \tuplet 3/2 {
      <a e' a cis e\RHd\RH>8
      \headsOff
      <a e' a cis e\RHu\RH>
      <a e' a cis e\RHu\RH>
      \headsOn
    }
  }
  \tupletsOff |
  <>\alzapúa
  \override Beam.positions = #'(2 . 2)
```

```
\score {
  \new StaffGroup <<
    \context Staff = "part" {
      \clef "G_8"
      \part
    }
    \context TabStaff {
      \part
    }
  >>
  \layout {
    ragged-right = ##t
  }
}
```

## Fret diagrams explained and developed

This snippet shows many possibilities for obtaining and tweaking fret diagrams.

&lt;&lt;

```

\chords {
  a1 a \bar "||" \break
  \repeat unfold 3 {
    c c c d d \bar "||" \break
  }
}

\new Voice {
  % Set global properties of fret diagram
  \override TextScript.size = 1.2
  \override TextScript.fret-diagram-details
    .finger-code = #'below-string
  \override TextScript.fret-diagram-details
    .dot-color = #'black

  % 1
  %
  % A chord for ukulele.
  a'1^\markup
    \override #'(fret-diagram-details
      . ((string-count . 4)
        (dot-color . white)
        (finger-code . in-dot)))
    \fret-diagram "4-2-2;3-1-1;2-o;1-o;"

  % 2
  %
  % A chord for ukulele, with formatting defined in definition
  % string: 1.2 * size, 4 strings, 4 frets, fingerings below,
  % string dot radius .35 of fret spacing, dot position 0.55 of
  % fret spacing.
  a'1^\markup
    \override #'(fret-diagram-details
      . ((dot-color . white)
        (open-string . "o")))
    \fret-diagram
      "s:1.2;w:4;h:3;f:2;d:0.35;p:0.55;4-2-2;3-1-1;2-o;1-o;"

  %%
  %% These chords will be in normal orientation
  %%

  % 3
  %
  % C major for guitar, barred on third fret: verbose style,
  % roman fret label, finger labels below string, straight barre.
  c'1^\markup
    % 110% of default size
    \override #'(size . 1.1)
    \override #'(fret-diagram-details

```

```

      . ((number-type . roman-lower)
        (finger-code . below-string)
        (barre-type . straight)))
\ fret-diagram-verbose #'((mute 6)
                          (place-fret 5 3 1)
                          (place-fret 4 5 2)
                          (place-fret 3 5 3)
                          (place-fret 2 5 4)
                          (place-fret 1 3 1)
                          (barre 5 1 3))

% 4
%
% C major for guitar, barred on third fret: double barre used
% to test barre function, verbose style.
c'1^\markup
% 110% of default size
\override #'(size . 1.1)
\override #'(fret-diagram-details
  . ((number-type . arabic)
    (dot-label-font-mag . 0.9)
    (finger-code . in-dot)
    (fret-label-font-mag . 0.6)
    (fret-label-vertical-offset . 0)
    (label-dir . -1)
    (mute-string . "M")
    (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 4 2 5)
                          (barre 5 1 3))

% 5
%
% C major for guitar, with capo on third fret: verbose style.
c'1^\markup
% 110% of default size
\override #'(size . 1.1)
\override #'(fret-diagram-details
  . ((number-type . roman-upper)
    (dot-label-font-mag . 0.9)
    (finger-code . none)
    (fret-label-vertical-offset . 0.5)
    (xo-font-magnification . 0.4)
    (xo-padding . 0.3)))
\ fret-diagram-verbose #'((mute 6)
                          (capo 3)

```

```

                                (open 5)
                                (place-fret 4 5 1)
                                (place-fret 3 5 2)
                                (place-fret 2 5 3)
                                (open 1))

% 6
%
% Simple D chord.
d'1^\markup
  \override #'(fret-diagram-details
    . ((finger-code . below-string)
      (dot-radius . 0.35)
      (string-thickness-factor . 0.3)
      (dot-position . 0.5)
      (fret-count . 3)))
  \fret-diagram-terse "x;x;o;2-1;3-2;2-3;"

% 7
%
% Simple D chord, large top fret thickness.
d'1^\markup
  \override #'(fret-diagram-details
    . ((finger-code . below-string)
      (dot-radius . 0.35)
      (dot-position . 0.5)
      (top-fret-thickness . 7)
      (fret-count . 3)))
  \fret-diagram-terse "x;x;o;2-1;3-2;2-3;"

%%
%% These chords will be in landscape orientation
%%
\override TextScript.fret-diagram-details
  .orientation = #'landscape

% 8
%
% C major for guitar, barred on third fret: verbose style,
% roman fret label, finger labels below string, straight
% barre.
c'1^\markup
  % 110% of default size
  \override #'(size . 1.1)
  \override #'(fret-diagram-details
    . ((number-type . roman-lower)
      (finger-code . below-string)
      (barre-type . straight)))
  \fret-diagram-verbose #'(mute 6)
                        (place-fret 5 3 1)

```

```

        (place-fret 4 5 2)
        (place-fret 3 5 3)
        (place-fret 2 5 4)
        (place-fret 1 3 1)
        (barre 5 1 3))

% 9
%
% C major for guitar, barred on third fret: Double barre
% used to test barre function, verbose style.
c'1~\markup
  % 110% of default size
  \override #'(size . 1.1)
  \override #'(fret-diagram-details
    . ((number-type . arabic)
      (dot-label-font-mag . 0.9)
      (finger-code . in-dot)
      (fret-label-font-mag . 0.6)
      (fret-label-vertical-offset . 0)
      (label-dir . -1)
      (mute-string . "M")
      (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 4 2 5)
    (barre 5 1 3))

% 10
%
% C major for guitar, with capo on third fret: verbose style.
c'1~\markup
  % 110% of default size
  \override #'(size . 1.1)
  \override #'(fret-diagram-details
    . ((number-type . roman-upper)
      (dot-label-font-mag . 0.9)
      (finger-code . none)
      (fret-label-vertical-offset . 0.5)
      (xo-font-magnification . 0.4)
      (xo-padding . 0.3)))
  \fret-diagram-verbose #'((mute 6)
    (capo 3)
    (open 5)
    (place-fret 4 5 1)
    (place-fret 3 5 2)
    (place-fret 2 5 3)
    (open 1))

```

```

% 11
%
% Simple D chord.
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;"

% 12
%
% Simple D chord, large top fret thickness.
d'1^\markup
  \override #'(fret-diagram-details
    . ((finger-code . below-string)
      (dot-radius . 0.35)
      (dot-position . 0.5)
      (top-fret-thickness . 7)
      (fret-count . 3)))
  \fret-diagram-terse "x;x;o;2-1;3-2;2-3;"

%%
%% These chords will be in opposing-landscape orientation.
%%
\override TextScript.fret-diagram-details
  .orientation = #'opposing-landscape

% 13
%
% C major for guitar, barred on third fret: verbose style,
% roman fret label, finger labels below string, straight
% barre.
c'1^\markup
  % 110% of default size
  \override #'(size . 1.1)
  \override #'(fret-diagram-details
    . ((number-type . roman-lower)
      (finger-code . below-string)
      (barre-type . straight)))
  \fret-diagram-verbose #'((mute 6)
    (place-fret 5 3 1)
    (place-fret 4 5 2)
    (place-fret 3 5 3)
    (place-fret 2 5 4)
    (place-fret 1 3 1)
    (barre 5 1 3))

```



```

% 14
%
% C major for guitar, barred on third fret: double barre
% used to test barre function, verbose style.
c'1^\markup
  % 110% of default size
  \override #'(size . 1.1)
  \override #'(fret-diagram-details
    . ((number-type . arabic)
      (dot-label-font-mag . 0.9)
      (finger-code . in-dot)
      (fret-label-font-mag . 0.6)
      (fret-label-vertical-offset . 0)
      (label-dir . -1)
      (mute-string . "M")
      (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 4 2 5)
    (barre 5 1 3))

% 15
%
% C major for guitar, with capo on third fret: verbose style.
c'1^\markup
  % 110% of default size
  \override #'(size . 1.1)
  \override #'(fret-diagram-details
    . ((number-type . roman-upper)
      (dot-label-font-mag . 0.9)
      (finger-code . none)
      (fret-label-vertical-offset . 0.5)
      (xo-font-magnification . 0.4)
      (xo-padding . 0.3)))
  \fret-diagram-verbose #'((mute 6)
    (capo 3)
    (open 5)
    (place-fret 4 5 1)
    (place-fret 3 5 2)
    (place-fret 2 5 3)
    (open 1))

% 16
%
% Simple D chord.
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;"

% 17
%
% Simple D chord, large top fret thickness.
d'1^\markup
  \override #'(fret-diagram-details
    . ((finger-code . below-string)
      (dot-radius . 0.35)
      (dot-position . 0.5)
      (top-fret-thickness . 7)
      (fret-count . 3)))
  \fret-diagram-terse "x;x;o;2-1;3-2;2-3;"
}
>>

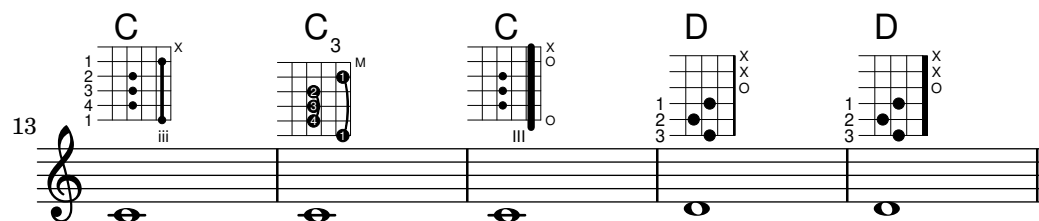
\paper {
  ragged-right = ##t
  system-system-spacing.basic-distance = 20
}

\layout {
  \context {
    \Score
    \override SpacingSpanner.spacing-increment = 3
  }
}

```

The image displays a musical score for guitar, organized into three systems. Each system contains a staff with a treble clef and a key signature of one sharp (F#). The score is annotated with fret diagrams and chord names (A, C, D) above the notes.

- System 1:** Features two measures. The first measure has a whole note with a fret diagram for an A chord (open strings, 2nd fret on the 4th string, 1st fret on the 3rd string). The second measure has a whole note with a fret diagram for an A chord (open strings, 2nd fret on the 4th string, 1st fret on the 3rd string, 1st fret on the 2nd string).
- System 2:** Features five measures. The first measure has a whole note with a fret diagram for a C chord (x, 3rd fret on the 1st string, 2nd fret on the 2nd string, 3rd fret on the 3rd string, 4th fret on the 4th string, 1st fret on the 5th string). The second measure has a whole note with a fret diagram for a C chord (x, 3rd fret on the 1st string, 2nd fret on the 2nd string, 3rd fret on the 3rd string, 4th fret on the 4th string, 1st fret on the 5th string). The third measure has a whole note with a fret diagram for a C chord (x, 3rd fret on the 1st string, 2nd fret on the 2nd string, 3rd fret on the 3rd string, 4th fret on the 4th string, 1st fret on the 5th string). The fourth measure has a whole note with a fret diagram for a D chord (x, x, 2nd fret on the 1st string, 3rd fret on the 2nd string, 2nd fret on the 3rd string, 3rd fret on the 4th string). The fifth measure has a whole note with a fret diagram for a D chord (x, x, 2nd fret on the 1st string, 3rd fret on the 2nd string, 2nd fret on the 3rd string, 3rd fret on the 4th string).
- System 3:** Features five measures. The first measure has a whole note with a fret diagram for a C chord (x, 3rd fret on the 1st string, 2nd fret on the 2nd string, 3rd fret on the 3rd string, 4th fret on the 4th string, 1st fret on the 5th string). The second measure has a whole note with a fret diagram for a C chord (x, 3rd fret on the 1st string, 2nd fret on the 2nd string, 3rd fret on the 3rd string, 4th fret on the 4th string, 1st fret on the 5th string). The third measure has a whole note with a fret diagram for a C chord (x, 3rd fret on the 1st string, 2nd fret on the 2nd string, 3rd fret on the 3rd string, 4th fret on the 4th string, 1st fret on the 5th string). The fourth measure has a whole note with a fret diagram for a D chord (x, x, 2nd fret on the 1st string, 3rd fret on the 2nd string, 2nd fret on the 3rd string, 3rd fret on the 4th string). The fifth measure has a whole note with a fret diagram for a D chord (x, x, 2nd fret on the 1st string, 3rd fret on the 2nd string, 2nd fret on the 3rd string, 3rd fret on the 4th string).



## 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 |
    }
  >>
}
>>

```

The image displays six chords (C, D, C, D, C, D) on a staff. Above each chord is a tablature diagram showing fingerings. The first three chords are labeled 'Default table', the next two 'New table from empty', and the last one 'New table from default'.

## Fretted-string harmonics in tablature

The following demonstrates fretted-string harmonics in a tablature.

```

pinchedHarmonics = {
  \textSpannerDown
  \override TextSpanner.bound-details.left.text =
    \markup { \halign #-0.5 \teeny "PH" }
  \override TextSpanner.style = #'dashed-line
  \override TextSpanner.dash-period = 0.6
  \override TextSpanner.bound-details.right.attach-dir = 1
  \override TextSpanner.bound-details.right.text =
    \markup { \draw-line #'(0 . 1) }
  \override TextSpanner.bound-details.right.padding = -0.5
}

harmonics = {
  % artificial harmonics (AH)
  \textLengthOn
  <\parenthesize b b'\harmonic>4_\markup { \teeny "AH 16" }
  <\parenthesize g g'\harmonic>4_\markup { \teeny "AH 17" }
  <\parenthesize d' d'\harmonic>2_\markup { \teeny "AH 19" }

  % pinched harmonics (PH)
  \pinchedHarmonics
  <a'\harmonic>2\startTextSpan

```

```

<d''\harmonic>4
<e'\harmonic>4\stopTextSpan

% tapped harmonics (TH)
<\parenthesize g\4 g'\harmonic>4_\markup { \teeny "TH 17" }
<\parenthesize a\4 a'\harmonic>4_\markup { \teeny "TH 19" }
<\parenthesize c'\3 c''\harmonic>2_\markup { \teeny "TH 17" }

% touch harmonics (TCH)
a4( <e''\harmonic>2. )_\markup { \teeny "TCH" }
}

frettedStrings = {
  % artificial harmonics (AH)
  \harmonicByFret 4 g4\3
  \harmonicByFret 5 d4\4
  \harmonicByFret 7 g2\3

  % pinched harmonics (PH)
  \harmonicByFret 7 d2\4
  \harmonicByFret 5 d4\4
  \harmonicByFret 7 a4\5

  % tapped harmonics (TH)
  \harmonicByFret 5 d4\4
  \harmonicByFret 7 d4\4
  \harmonicByFret 5 g2\3

  % touch harmonics (TCH)
  a4 \harmonicByFret 9 g2.\3
}

\score {
  <<
    \new Staff
    \with { \omit StringNumber } {
      \new Voice {
        \clef "treble_8"
        \harmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \frettedStrings
      }
    }
  >>
}

```

## Guitar slides

Unlike glissandos, slides may go from an imprecise point of the fretboard to a specific fret. A good way to do this is to add a hidden grace note before the note which is actually played, as demonstrated in the following example.

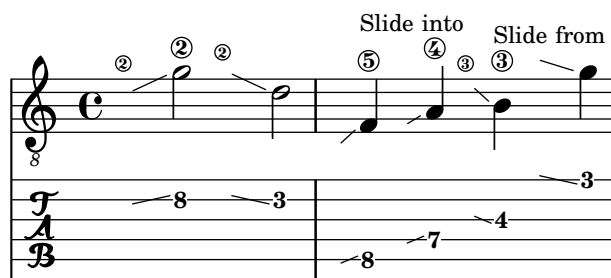
*% Hide fret number: useful to draw slide into/from a casual point of  
% the fretboard.*

```
hideFretNumber = {
  \once \hide TabNoteHead
  \once \hide NoteHead
  \once \omit Stem
  \once \omit Flag
  \once \override NoteHead.no-ledgers = ##t
  \once \override Glissando.bound-details.left.padding = #0.3
}

music= \relative c' {
  \grace { \hideFretNumber d8\2 \glissando s2 } g2\2
  \grace { \hideFretNumber g8\2 \glissando s2 } d2 |

  \grace { \hideFretNumber c,8 \glissando s }
    f4\5~\markup \tiny { Slide into }
  \grace { \hideFretNumber f8 \glissando s } a4\4
  \grace { \hideFretNumber e'8\3 \glissando s }
    b4\3~\markup \tiny { Slide from }
  \grace { \hideFretNumber b'8 \glissando s2 } g4 |
}

\score {
  <<
    \new Staff {
      \clef "G_8"
      \music
    }
    \new TabStaff {
      \music
    }
  >>
}
```



## Guitar strum rhythms

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

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

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

The image shows a musical score for a song. The top staff is a guitar fretboard with four diagrams for chords: C (x o o), F (1 3 4 2 1 1), G (2 1 3), and C (x o o). The bottom staff is a vocal line with the lyrics: "This is my song. I like to sing." The notes are: This (C4), is (E4), my (G4), song. (A4), I (C5), like (E4), to (G4), sing. (A4).

## Hammer-on and pull-off

Hammer-on and pull-off can be obtained using slurs.

```
\new TabStaff {
  \relative c' {
    d4( e\2)
    a( g)
  }
}
```

The diagram shows a guitar fretboard with a slur over the notes 3, 5, 5, 3. The notes are labeled with their fret numbers: 3, 5, 5, 3.

## Hammer-on and pull-off using chords

When using hammer-on or pull-off with chorded notes, only a single arc is drawn. However “double arcs” are possible by setting the `doubleSlurs` property to `#t`.

```
\new TabStaff {
  \relative c' {
    % chord hammer-on and pull-off
    \set doubleSlurs = ##t
    <g' b>8( <a c> <g b>)
  }
}
```

The diagram shows a guitar fretboard with a double arc over the notes 7-8-7 and 8-10-8. The notes are labeled with their fret numbers: 7-8-7 and 8-10-8.

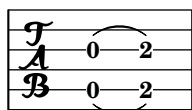
## Hammer-on and pull-off using voices

The arc of hammer-on and pull-off is upwards in voices one and three and downwards in voices two and four:

```
\new TabStaff {
  \relative c' {
    << { \voiceOne g2( a) }
    \\ { \voiceTwo a,( b) }
    >> \oneVoice
  }
}
```



}



## How to change fret diagram position

If you want to move the position of a fret diagram, for example, to avoid collision, or to place it between two notes, you have various possibilities.

- 1) Modify the value of the padding or extra-offset property (as shown in the first line).
- 2) You can add an invisible voice and attach the fret diagrams to the invisible notes in that voice (as shown in the second line).

If you need to move the fret according with a rhythmic position inside the bar (in the example, the third beat of the measure) the second example is better, because the fret is aligned with the third beat itself.

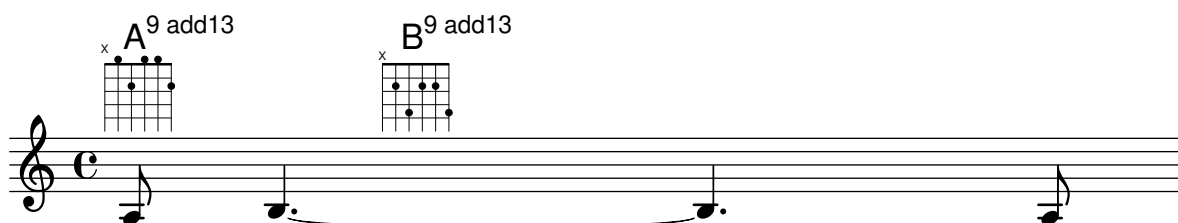
```

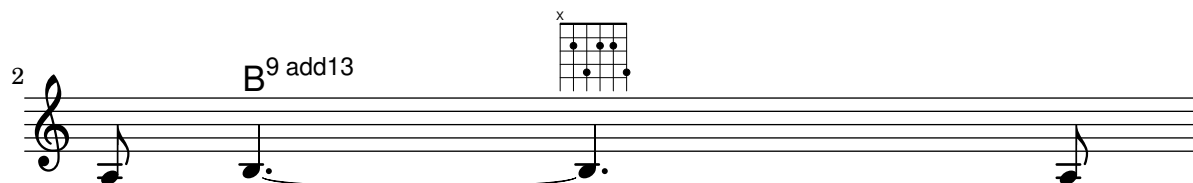
harmonies = \chordmode
{
  a8:13
  \once \override ChordNames.ChordName.extra-offset = #'(10 . 0)
  b8:13 s4. |
  s2 b2:13
}

\score {
  <<
    \new ChordNames \harmonies
    \new Staff {
      % Method 1.
      a8^\markup \fret-diagram "6-x;5-0;4-2;3-0;2-0;1-2;"
      \once \override TextScript.extra-offset = #'(10 . 0)
      b4.~^\markup \fret-diagram "6-x;5-2;4-4;3-2;2-2;1-4;"
      b4. a8 | \break

      % Method 2.
      <<
        { a8 b4.~ b4. a8 }
        { s2 s2^\markup \fret-diagram "6-x;5-2;4-4;3-2;2-2;1-4;" }
      >> |
    }
  >>
}

```





## Jazz combo template

This is quite an advanced template, for a jazz ensemble. Note that all instruments use `\key c \major`. This refers to the key in concert pitch; the key will be automatically transposed if the music is within a `\transpose` section.

```
\header {
  title = "Song"
  subtitle = "(tune)"
  composer = "Me"
  meter = "moderato"
  piece = "Swing"
  tagline = \markup \column {
    "LilyPond example file by Amelie Zapf,"
    "Berlin 07/07/2003" }
}

% To make the example display properly in the documentation.
\paper {
  paper-width = 130\mm
  paper-height = 205\mm
}

% #(set-global-staff-size 16)

\include "english.ly"

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Some macros %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sl = { \override NoteHead.style = #'slash
       \hide Stem }
nsl = { \revert NoteHead.style
        \undo \hide Stem }
crOn = \override NoteHead.style = #'cross
crOff = \revert NoteHead.style

% Insert chord name style stuff here.

jazzChords = { }

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Keys'n'things %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global = { \time 4/4 }

Key = { \key c \major }
```

```

% ##### Horns #####

% ----- Trumpet -----
trpt = \transpose c d \relative c' {
  \Key
  c1 | c | c |
}
trpHarmony = \transpose c' d {
  \jazzChords
}
trumpet = {
  \global
  \clef treble
  \trpt
}

% ----- Alto Saxophone -----
alto = \transpose c a \relative c' {
  \Key
  c1 | c | c |
}
altoHarmony = \transpose c' a {
  \jazzChords
}
altoSax = {
  \global
  \clef treble
  \alto
}

% ----- Baritone Saxophone -----
bari = \transpose c a' \relative c {
  \Key
  c1 | c1 |
  \sl d4^"Solo" d d d \ns1 |
}
bariHarmony = \transpose c' a \chordmode {
  \jazzChords
  s1 | s |
  d2:maj e:m7 |
}
bariSax = {
  \global
  \clef treble
  \bari
}

% ----- Trombone -----
tbone = \relative c {
  \Key
  c1 | c | c |
}

```

```

}
tboneHarmony = \chordmode {
  \jazzChords
}
trombone = {
  \global
  \clef bass
  \tbone
}

% ##### Rhythm Section #####

% ----- Guitar -----
gtr = \relative c'' {
  \Key
  c1 |
  \sl b4 b b b \ns1 |
  c1 |
}
gtrHarmony = \chordmode {
  \jazzChords
  s1 | c2:min7+ d2:maj9 | s1 |
}
guitar = {
  \global
  \clef treble
  \gtr
}

%% ----- Piano -----
rhUpper = \relative c'' {
  \voiceOne
  \Key
  c1 | c | c |
}
rhLower = \relative c' {
  \voiceTwo
  \Key
  e1 | e | e |
}

lhUpper = \relative c' {
  \voiceOne
  \Key
  g1 | g | g |
}
lhLower = \relative c {
  \voiceTwo
  \Key
  c1 | c | c |
}

```

```

PianoRH = {
  \clef treble
  \global
  <<
    \new Voice = "one" \rhUpper
    \new Voice = "two" \rhLower
  >>
}
PianoLH = {
  \clef bass
  \global
  <<
    \new Voice = "one" \lhUpper
    \new Voice = "two" \lhLower
  >>
}

piano = <<
  \new Staff = "upper" \PianoRH
  \new Staff = "lower" \PianoLH
>>

% ----- Bass Guitar -----
Bass = \relative c {
  \Key
  c1 | c | c |
}
bass = {
  \global
  \clef bass
  \Bass
}

% ----- Drums -----
up = \drummode {
  \voiceOne
  hh4 <hh sn> hh <hh sn> |
  hh4 <hh sn> hh <hh sn> |
  hh4 <hh sn> hh <hh sn> |
}
down = \drummode {
  \voiceTwo
  bd4 s bd s |
  bd4 s bd s |
  bd4 s bd s |
}

drumContents = {
  \global
  <<
    \new DrumVoice \up

```

```

\new DrumVoice \down
>>
}

%%%%%%%%%% It All Goes Together Here %%%%%%%%%%%%%%

\book { % For the LilyPond documentation.
\score {
  <<
    \new StaffGroup = "horns" <<
      \new Staff = "trumpet" \with { instrumentName = "Trumpet" }
      \trumpet
      \new Staff = "altosax" \with { instrumentName = "Alto Sax" }
      \altoSax
      \new ChordNames = "barichords" \with { instrumentName = "Bari Sax" }
      \bariHarmony
      \new Staff = "barisax" \with { instrumentName = "Bari Sax" }
      \bariSax
      \new Staff = "trombone" \with { instrumentName = "Trombone" }
      \trombone
    >>

    \new StaffGroup = "rhythm" <<
      \new ChordNames = "chords" \with { instrumentName = "Guitar" }
      \gtrHarmony
      \new Staff = "guitar" \with { instrumentName = "Guitar" }
      \guitar
      \new PianoStaff = "piano" \with {
        instrumentName = "Piano"
        midiInstrument = "acoustic grand"
      } \piano
      \new Staff = "bass" \with { instrumentName = "Bass" }
      \bass
      \new DrumStaff \with { instrumentName = "Drums" }
      \drumContents
    >>
  >>

  \layout {
    \context {
      \Staff
      \RemoveEmptyStaves
    }
    \context {
      \Score
      \override BarNumber.padding = 3
      \override RehearsalMark.padding = 2
      skipBars = ##t
    }
  }
  \midi { }
}

```

}

## Song

(tune)

Me

moderato  
Swing

Trumpet

Alto Sax

Bari Sax

Trombone

Guitar

Piano

Bass

Drums

B $\Delta$  Solo C $\sharp$ m<sup>7</sup>

Cm $\Delta$  D $\Delta$ <sup>9</sup>

LilyPond example file by Amelie Zapf,  
Berlin 07/07/2003

## Laissez vibrer ties

Laissez vibrer ties have a fixed size. Their positioning can be tuned using the tie-configuration property.

See also snippet “Longer laissez vibrer ties”.

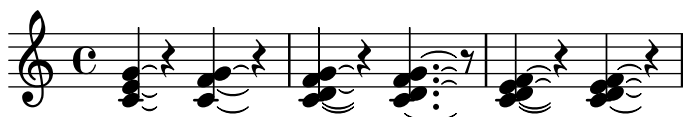
```
\relative c' {
  <c e g>4\laissezVibrer r <c f g>\laissezVibrer r
  <c d f g>4\laissezVibrer r <c d f g>4.\laissezVibrer r8

  <c d e f>4\laissezVibrer r
```

```

\override LaissezVibrerTieColumn.tie-configuration
  = #`((-7 . ,DOWN)
        (-5 . ,DOWN)
        (-3 . ,UP)
        (-1 . ,UP))
<c d e f>4\laissezVibrer r
}

```



## Let TabStaff print the topmost string at bottom

In tablatures, the first string is usually printed topmost. If you want to have it at the bottom, set the `stringOneTopmost` context property to `##f`. For a context-wide setting this could be done in the `\layout` block as well.

```

%\layout {
% \context {
%   \Score
%   stringOneTopmost = ##f
% }
% \context {
%   \TabStaff
%   tablatureFormat = #fret-letter-tablature-format
% }
%}

m = {
  \cadenzaOn
  e, b, e gis! b e'
  \bar "||"
}

<<
\new Staff {
  \clef "G_8"
  <>_"default" \m
  <>_"italian (historic)"\m
}
\new TabStaff
{
  \m
  \set Score.stringOneTopmost = ##f
  \set TabStaff.tablatureFormat = #fret-letter-tablature-format
  \m
}
>>

```



default                      italian (historic)

## Letter tablature formatting

Tablature can be formatted using letters instead of numbers.

```
music = \relative c {
  c4 d e f
  g4 a b c
  d4 e f g
}

<<
\new Staff {
  \clef "G_8"
  \music
}
\new TabStaff \with {
  tablatureFormat = #fret-letter-tablature-format
} {
  \music
}
>>
```

## Open-string harmonics in tablature

This snippet demonstrates open-string harmonics.

```
openStringHarmonics = {
  \textSpannerDown
  \override TextSpanner.staff-padding = 3
  \override TextSpanner.dash-fraction = 0.3
  \override TextSpanner.dash-period = 1

  % first harmonic
  \override TextSpanner.bound-details.left.text =
    \markup\small "1st harm. "
  \harmonicByFret 12 e,2\6\startTextSpan
  \harmonicByRatio #1/2 e,\6\stopTextSpan
```

```
% second harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "2nd harm. "
\harmonicByFret 7 e,\6\startTextSpan
\harmonicByRatio #1/3 e,\6
\harmonicByFret 19 e,\6
\harmonicByRatio #2/3 e,\6\stopTextSpan
%\harmonicByFret 19 < e,\6 a,\5 d\4 >
%\harmonicByRatio #2/3 < e,\6 a,\5 d\4 >
```

```
% third harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "3rd harm. "
\harmonicByFret 5 e,\6\startTextSpan
\harmonicByRatio #1/4 e,\6
\harmonicByFret 24 e,\6
\harmonicByRatio #3/4 e,\6\stopTextSpan
\break
```

```
% fourth harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "4th harm. "
\harmonicByFret 4 e,\6\startTextSpan
\harmonicByRatio #1/5 e,\6
\harmonicByFret 9 e,\6
\harmonicByRatio #2/5 e,\6
\harmonicByFret 16 e,\6
\harmonicByRatio #3/5 e,\6\stopTextSpan
```

```
% fifth harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "5th harm. "
\harmonicByFret 3 e,\6\startTextSpan
\harmonicByRatio #1/6 e,\6\stopTextSpan
\break
```

```
% sixth harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "6th harm. "
\harmonicByFret 2.7 e,\6\startTextSpan
\harmonicByRatio #1/7 e,\6\stopTextSpan
```

```
% seventh harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "7th harm. "
\harmonicByFret 2.3 e,\6\startTextSpan
\harmonicByRatio #1/8 e,\6\stopTextSpan
```

```
% eighth harmonic
\override TextSpanner.bound-details.left.text =
  \markup\small "8th harm. "
\harmonicByFret 2 e,\6\startTextSpan
```

```

\harmonicByRatio #1/9 e,\6\stopTextSpan
}

\score {
  <<
    \new Staff \with { \omit StringNumber } {
      \new Voice {
        \clef "treble_8"
        \openStringHarmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \openStringHarmonics
      }
    }
  >>
}

```

The image displays a musical score for the first six harmonics of the E string. The score is organized into three systems, each containing a musical staff and a corresponding tablature staff. The first system covers the 1st, 2nd, and 3rd harmonics, with fret numbers (12), (12), (7), (7), (19), (19), (5), (5), (24), and (24) shown in the tablature. The second system covers the 4th and 5th harmonics, with fret numbers (4), (4), (9), (9), (16), (16), (3), and (3). The third system covers the 6th, 7th, and 8th harmonics, with fret numbers (2.7), (2.7), (2.3), (2.3), (2), and (2). The musical staff shows the notes for each harmonic, with the 1st harmonic being an E4, the 2nd being an E5, and the 3rd being an E6. The 4th harmonic is a G5, the 5th is an A5, the 6th is a B5, the 7th is a C6, and the 8th is a D6. The tablature staff shows the fret numbers for each harmonic, with the 1st harmonic being (12), the 2nd being (12), the 3rd being (7), the 4th being (7), the 5th being (19), the 6th being (19), the 7th being (5), the 8th being (5), the 9th being (24), and the 10th being (24).

## 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.

```

#(define RH rightHandFinger)

```

```

\relative c {
  \clef "treble_8"

```

```

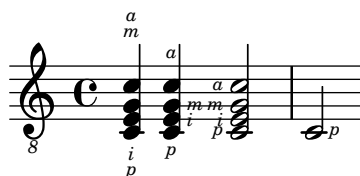
\set strokeFingerOrientations = #'(up down)
<c\RH 1 e\RH 2 g\RH 3 c\RH 4 >4

\set strokeFingerOrientations = #'(up right down)
<c\RH 1 e\RH 2 g\RH 3 c\RH 4 >4

\set strokeFingerOrientations = #'(left)
<c\RH 1 e\RH 2 g\RH 3 c\RH 4 >2

\set strokeFingerOrientations = #'(right)
c\RH 1
}

```



## Polyphony in tablature

Polyphony is created the same way in a TabStaff as in a regular staff.

```

upper = \relative c' {
  \time 12/8
  \key e \minor
  \voiceOne
  r4. r8 e, fis g16 b g e e' b c b a g fis e
}

lower = \relative c {
  \key e \minor
  \voiceTwo
  r16 e d c b a g4 fis8 e fis g a b c
}

\score {
  \new StaffGroup = "tab with traditional" <<
    \new Staff = "guitar traditional" <<
      \clef "treble_8"
      \new Voice = "upper" \upper
      \new Voice = "lower" \lower
    >>

    \new TabStaff = "guitar tab" <<
      \new TabVoice = "upper" \upper
      \new TabVoice = "lower" \lower
    >>
  >>
}

```

## Setting up predefined fretboards for other instruments

Predefined fret diagrams can be added for new instruments in addition to the standard diagrams 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 be \included into each of your compositions.
```

```
cuatroTuning = #`((ly:make-pitch 0 6 0)
                  ,(ly:make-pitch 1 3 SHARP)
                  ,(ly:make-pitch 1 1 0)
                  ,(ly:make-pitch 0 5 0))
```

```
dSix = { <a\4 b\1 d\3 fis\2> }
dMajor = { <a\4 d\1 d\3 fis \2> }
aMajSeven = { <a\4 cis\1 e\3 g\2> }
dMajSeven = { <a\4 c\1 d\3 fis\2> }
gMajor = { <b\4 b\1 d\3 g\2> }
```

```
\storePredefinedDiagram #default-fret-table \dSix
                        #cuatroTuning
                        "o;o;o;o;"
\storePredefinedDiagram #default-fret-table \dMajor
                        #cuatroTuning
                        "o;o;o;3-3;"
\storePredefinedDiagram #default-fret-table \aMajSeven
                        #cuatroTuning
                        "o;2-2;1-1;2-3;"
\storePredefinedDiagram #default-fret-table \dMajSeven
                        #cuatroTuning
                        "o;o;o;1-1;"
\storePredefinedDiagram #default-fret-table \gMajor
                        #cuatroTuning
                        "2-2;o;1-1;o;"
```

```

% End of potential include file `predefined-cuatro-fretboards.ly`.

#(set-global-staff-size 16)

primerosNames = \chordmode {
  d:6 d a:maj7 d:maj7
  g
}
primeros = {
  \dSix \dMajor \aMajSeven \dMajSeven
  \gMajor
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \primerosNames
    }

    \new Staff {
      \new Voice \with {
        \remove "New_fingering_engraver"
      }
      \relative c'' {
        \primeros
      }
    }

    \new FretBoards {
      \set Staff.stringTunings = #cuatroTuning
      % \override FretBoard
      % #'(fret-diagram-details string-count) = 4
      \override FretBoard.fret-diagram-details.finger-code = #'in-dot
      \primeros
    }

    \new TabStaff \relative c'' {
      \set TabStaff.stringTunings = #cuatroTuning
      \primeros
    }

  >>

  \layout {
    \context {
      \Score
      \override SpacingSpanner.base-shortest-duration =
        \musicLength 16
    }
  }
}

```

```
\midi { }
}
```

The image shows a musical score for a guitar piece. The top staff is a treble clef with a common time signature. It shows five chords: D<sup>6</sup>, D, A<sup>Δ</sup>, D<sup>Δ</sup>, and G. Below the staff are five guitar fretboard diagrams corresponding to these chords. At the bottom, there are three staves: a treble clef staff, a bass clef staff, and a double bass clef staff, each with a sequence of numbers representing fret positions for the strings.

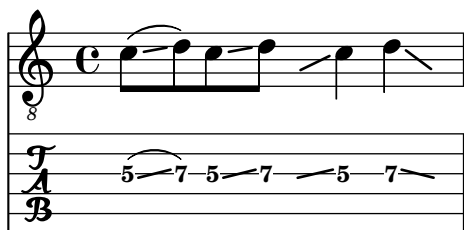
## Slides in tablature

Slides can be typeset in both Staff and TabStaff contexts.

```
slides = {
  c'8\3(\glissando d'8\3)
  c'8\3\glissando d'8\3
  \hideNotes
  \grace { g16\glissando }
  \unHideNotes
  c'4\3
  \afterGrace d'4\3\glissando {
    \stemDown \hideNotes
    g16 }
  \unHideNotes
}

\score {
  <<
    \new Staff { \clef "treble_8" \slides }
    \new TabStaff { \slides }
  >>

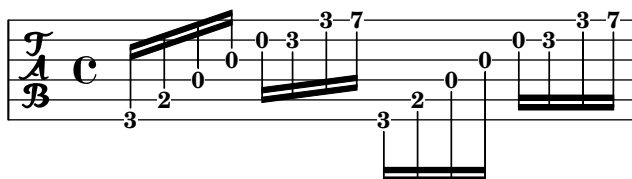
  \layout {
    \context {
      \Score
      \override Glissando.minimum-length = 4
      \override Glissando.springs-and-rods =
        #ly:spanner::set-spacing-rods
      \override Glissando.thickness = 2
      \omit StringNumber
      % or:
      \%override StringNumber.stencil = ##f
    }
  }
}
```



## Stem and beam behavior in tablature

The direction of stems is controlled the same way in tablature as in traditional notation. Beams can be made horizontal, as shown in this example.

```
\new TabStaff {
  \relative c {
    \tabFullNotation
    g16 b d g b d g b
    \stemDown
    \override Beam.concaveness = 10000
    g,,16 b d g b d g b
  }
}
```



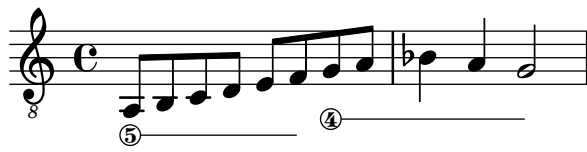
## String number extender lines

Make an extender line for string number indications, showing that a series of notes is supposed to be played all on the same string.

```
stringNumberSpanner =
  #(define-music-function (StringNumber) (string?)
    #{
      \override TextSpanner.style = #'solid
      \override TextSpanner.font-size = #-5
      \override TextSpanner.bound-details.left.stencil-align-dir-y = #CENTER
      \override TextSpanner.bound-details.left.text =
        \markup { \circle \number $StringNumber }
    #})

\relative c {
  \clef "treble_8"
  \textSpannerDown
  \stringNumberSpanner "5" a8\startTextSpan b c d
  e f\stopTextSpan \stringNumberSpanner "4" g\startTextSpan a |
  bes4 a g2\stopTextSpan
}
```





## 13 Percussion

See also Section “Percussion” in *Notation Reference*.

### Adding drum parts

Using the powerful pre-configured tools such as the `\drummode` function and the `DrumStaff` context, inputting drum parts is quite easy: drums are placed at their own staff positions (with a special clef symbol) and have note heads according to the drum. Attaching an extra symbol to the drum or restricting the number of lines is possible.

```

drh = \drummode {
  cymc4.^"crash" hhc16^"h.h." hh hhc8 hho hhc8 hh16 hh
  hhc4 r4 r2
}
drl = \drummode {
  bd4 sn8 bd bd4 << bd ss >>
  bd8 tommh tommh bd toml toml bd tomfh16 tomfh
}
timb = \drummode {
  timh4 ssh timl8 ssh r timh r4
  ssh8 timl r4 cb8 cb
}

\score {
  <<
    \new DrumStaff \with {
      instrumentName = "timbales"
      drumStyleTable = #timbales-style
      \override StaffSymbol.line-count = #2
      \override BarLine.bar-extent = #'(-1 . 1)
    }
    <<
      \timb
    >>
    \new DrumStaff \with { instrumentName = "drums" }
    <<
      \new DrumVoice { \stemUp \drh }
      \new DrumVoice { \stemDown \drl }
    >>
  >>
  \layout { }
  \midi { \tempo 4 = 120 }
}

```

The image displays two staves of musical notation. The top staff is labeled 'timbales' and features a C-clef and common time signature. It contains a series of notes and rests, with specific drum sounds indicated by text labels: 'crash' (marked with a cross), 'h.h.' (marked with a plus sign), and various rhythmic patterns marked with 'x' and 'o'. The bottom staff is labeled 'drums' and also features a C-clef and common time signature. It contains a series of notes and rests, with specific drum sounds indicated by text labels: 'bd' (bass drum, marked with a cross), 'sn' (snare drum, marked with a plus sign), and various rhythmic patterns marked with 'x' and 'o'.

## Cow and ride bell example

Two different bells, entered with ‘cb’ (cow bell) and ‘rb’ (ride bell).

```
#(define mydrums '((ridebell default #f 3)
                  (cowbell default #f -2)))

\new DrumStaff \with { instrumentName = #"Different Bells" }

\drummode {
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \set DrumStaff.clefPosition = 0.5
  \override DrumStaff.StaffSymbol.line-positions = #'(-2 3)
  \override Staff.BarLine.bar-extent = #'(-1.0 . 1.5)

  \time 2/4
  rb8 8 cb8 16 rb16-> ~ |
  16 8 16 cb8 8 |
}
```



## Heavily customized polymetric time signatures

Though the polymetric time signature shown is not the most essential item here, it has been included to show the beat of this piece (which is the template of a real Balkan song, by the way).

```
melody = \relative c'' {
  \key g \major
  \compoundMeter #'((3 8) (2 8) (2 8) (3 8) (2 8) (2 8)
                  (2 8) (2 8) (3 8) (2 8) (2 8))
  \set Timing.beamExceptions = #'()
  \set Timing.beatStructure = 3,2,2,3,2,2,2,2,3,2,2
  c8 c c d4 c8 c b c b a4 g fis8 e d c b' c d e4-^ fis8 g \break
  c,4. d4 c4 d4. c4 d c2 d4. e4-^ d4
  c4. d4 c4 d4. c4 d c2 d4. e4-^ d4 \break
}

drum = \new DrumStaff \drummode {
  \repeat volta 2 {
    bd4.^{\markup { Drums } sn4 bd \bar "}
    sn4. bd4 sn \bar "
    bd sn bd4. sn4 bd
  }
}

\new Staff {
  \melody
  \drum
}
```



## High and low woodblock example

Two Woodblocks, entered with 'wbh' (high woodblock) and 'wbl' (low woodblock). The length of the bar line has been altered with an `\override` command, otherwise it would be too short. The positions of the two staff lines also have to be explicitly defined.

```
% 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 #f 3)
                    (lowwoodblock default #f -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 = #'(-1.0 . 1.5)
  % small correction for the clef:
  \set DrumStaff.clefPosition = 0.5
}
```

```
\new DrumStaff {
  % with this you load your new drum style table
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)

  \woodstaff

  \drummode {
    \time 2/4
    wbh8 16 16 8-> 8 |
    wbl8 16 16-> ~ 16 16 r8 |
  }
}
```



## Jazz combo template

This is quite an advanced template, for a jazz ensemble. Note that all instruments use `\key c \major`. This refers to the key in concert pitch; the key will be automatically transposed if the music is within a `\transpose` section.

```
\header {
  title = "Song"
  subtitle = "(tune)"
  composer = "Me"
  meter = "moderato"
  piece = "Swing"
  tagline = \markup \column {
    "LilyPond example file by Amelie Zapf,"
    "Berlin 07/07/2003" }
}

% To make the example display properly in the documentation.
\paper {
  paper-width = 130\mm
  paper-height = 205\mm
}

% #(set-global-staff-size 16)

\include "english.ly"

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Some macros %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sl = { \override NoteHead.style = #'slash
      \hide Stem }
nsl = { \revert NoteHead.style
      \undo \hide Stem }
crOn = \override NoteHead.style = #'cross
crOff = \revert NoteHead.style

% Insert chord name style stuff here.

jazzChords = { }

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Keys'n'things %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global = { \time 4/4 }

Key = { \key c \major }

% ##### Horns #####

% ----- Trumpet -----
trpt = \transpose c d \relative c' {
  \Key
```

```

    c1 | c | c |
}
trpHarmony = \transpose c' d {
  \jazzChords
}
trumpet = {
  \global
  \clef treble
  \trpt
}

% ----- Alto Saxophone -----
alto = \transpose c a \relative c' {
  \Key
  c1 | c | c |
}
altoHarmony = \transpose c' a {
  \jazzChords
}
altoSax = {
  \global
  \clef treble
  \alto
}

% ----- Baritone Saxophone -----
bari = \transpose c a' \relative c {
  \Key
  c1 | c1 |
  \sl d4^"Solo" d d d \ns1 |
}
bariHarmony = \transpose c' a \chordmode {
  \jazzChords
  s1 | s |
  d2:maj e:m7 |
}
bariSax = {
  \global
  \clef treble
  \bari
}

% ----- Trombone -----
tbone = \relative c {
  \Key
  c1 | c | c |
}
tboneHarmony = \chordmode {
  \jazzChords
}
trombone = {
  \global

```

```

\clef bass
\tbone
}

% ##### Rhythm Section #####

% ----- Guitar -----
gtr = \relative c'' {
  \Key
  c1 |
  \sl b4 b b b \ns1 |
  c1 |
}
gtrHarmony = \chordmode {
  \jazzChords
  s1 | c2:min7+ d2:maj9 | s1 |
}
guitar = {
  \global
  \clef treble
  \gtr
}

%% ----- Piano -----
rhUpper = \relative c'' {
  \voiceOne
  \Key
  c1 | c | c |
}
rhLower = \relative c' {
  \voiceTwo
  \Key
  e1 | e | e |
}

lhUpper = \relative c' {
  \voiceOne
  \Key
  g1 | g | g |
}
lhLower = \relative c {
  \voiceTwo
  \Key
  c1 | c | c |
}

PianoRH = {
  \clef treble
  \global
  <<
  \new Voice = "one" \rhUpper
  \new Voice = "two" \rhLower

```

```

>>
}
PianoLH = {
  \clef bass
  \global
  <<
    \new Voice = "one" \lhUpper
    \new Voice = "two" \lhLower
  >>
}

piano = <<
  \new Staff = "upper" \PianoRH
  \new Staff = "lower" \PianoLH
>>

% ----- Bass Guitar -----
Bass = \relative c {
  \Key
  c1 | c | c |
}
bass = {
  \global
  \clef bass
  \Bass
}

% ----- Drums -----
up = \drummode {
  \voiceOne
  hh4 <hh sn> hh <hh sn> |
  hh4 <hh sn> hh <hh sn> |
  hh4 <hh sn> hh <hh sn> |
}
down = \drummode {
  \voiceTwo
  bd4 s bd s |
  bd4 s bd s |
  bd4 s bd s |
}

drumContents = {
  \global
  <<
    \new DrumVoice \up
    \new DrumVoice \down
  >>
}

%%%%%%%%%% It All Goes Together Here %%%%%%%%%%%

```



```

\book { % For the LilyPond documentation.
\score {
  <<
    \new StaffGroup = "horns" <<
      \new Staff = "trumpet" \with { instrumentName = "Trumpet" }
      \trumpet
      \new Staff = "altosax" \with { instrumentName = "Alto Sax" }
      \altoSax
      \new ChordNames = "barichords" \with { instrumentName = "Bari Sax" }
      \bariHarmony
      \new Staff = "barisax" \with { instrumentName = "Bari Sax" }
      \bariSax
      \new Staff = "trombone" \with { instrumentName = "Trombone" }
      \trombone
    >>

    \new StaffGroup = "rhythm" <<
      \new ChordNames = "chords" \with { instrumentName = "Guitar" }
      \gtrHarmony
      \new Staff = "guitar" \with { instrumentName = "Guitar" }
      \guitar
      \new PianoStaff = "piano" \with {
        instrumentName = "Piano"
        midiInstrument = "acoustic grand"
      } \piano
      \new Staff = "bass" \with { instrumentName = "Bass" }
      \bass
      \new DrumStaff \with { instrumentName = "Drums" }
      \drumContents
    >>
  >>

  \layout {
    \context {
      \Staff
      \RemoveEmptyStaves
    }
    \context {
      \Score
      \override BarNumber.padding = 3
      \override RehearsalMark.padding = 2
      skipBars = ##t
    }
  }
}
\midi { }
}

```

## Song

(tune)

Me

moderato  
Swing

Trumpet

Alto Sax

Bari Sax

Trombone

Guitar

Piano

Bass

Drums

$B^{\Delta}$   $C\sharp m^7$   
Solo

$Cm^{\Delta}$   $D^{\Delta 9}$

LilyPond example file by Amelie Zapf,  
Berlin 07/07/2003

## Percussion beaters

Graphic symbols for percussion instruments are not natively supported; however it is possible to include such symbols, either as an external EPS file or as embedded PostScript code inside a markup, as demonstrated in this example.

```
stick = \markup \with-dimensions #'(0.80 . 5.2) #'(0.85 . 5.2) {
  \postscript "
    0 6 translate
    0.8 -0.8 scale
    0 0 0 setrgbcolor
    [] 0 setdash
    1 setlinewidth
```

```

0 setlinejoin
0 setlinecap
gsave [1 0 0 1 0 0] concat
gsave [1 0 0 1 -3.5406095 -199.29342] concat
gsave
0 0 0 setrgbcolor
newpath
7.1434065 200.94354 moveto
7.2109628 200.90454 7.2785188 200.86554 7.3460747 200.82654 curveto
8.2056347 202.31535 9.0651946 203.80414 9.9247546 205.29295 curveto
9.8571989 205.33195 9.7896429 205.37095 9.7220864 205.40996 curveto
8.8625264 203.92115 8.0029664 202.43233 7.1434065 200.94354 curveto
closepath
eofill
grestore
gsave
0 0 0 setrgbcolor
newpath
4.9646672 203.10444 moveto
5.0036707 203.03688 5.0426744 202.96933 5.0816777 202.90176 curveto
6.5704792 203.76133 8.0592809 204.6209 9.5480824 205.48045 curveto
9.5090791 205.54801 9.4700754 205.61556 9.4310717 205.68311 curveto
7.94227 204.82356 6.4534687 203.96399 4.9646672 203.10444 curveto
closepath
eofill
grestore
gsave
<<
/ShadingType 3
/ColorSpace /DeviceRGB
/Coords [113.13708 207.87465 0 113.13708 207.87465 16.162441]
/Extend [true true]
/Domain [0 1]
/Function <<
/FunctionType 3
/Functions
[
<<
/FunctionType 2
/Domain [0 1]
/C0 [1 1 1]
/C1 [0.72941178 0.72941178 0.72941178]
/N 1
>>
]
/Domain [0 1]
/Bounds [ ]
/Encode [ 0 1 ]
>>
>>
newpath
7.6422017 200.76488 moveto

```

```

7.6505696 201.02554 7.3905363 201.24867 7.1341335 201.20075 curveto
6.8759501 201.16916 6.6949602 200.87978 6.7801462 200.63381 curveto
6.8480773 200.39155 7.1438307 200.25377 7.3728389 200.35861 curveto
7.5332399 200.42458 7.6444521 200.59122 7.6422017 200.76488 curveto
closepath
clip
gsave [
  0.052859054 0.063089841 -0.020912282 0.017521108 5.7334261 189.76443
] concat
shfill
grestore
grestore
0 0 0 setrgbcolor
[] 0 setdash
0.027282091 setlinewidth
0 setlinejoin
0 setlinecap
newpath
7.6422017 200.76488 moveto
7.6505696 201.02554 7.3905363 201.24867 7.1341335 201.20075 curveto
6.8759501 201.16916 6.6949602 200.87978 6.7801462 200.63381 curveto
6.8480773 200.39155 7.1438307 200.25377 7.3728389 200.35861 curveto
7.5332399 200.42458 7.6444521 200.59122 7.6422017 200.76488 curveto
closepath
stroke
gsave
<<
/ShadingType 3
/ColorSpace /DeviceRGB
/Coords [113.13708 207.87465 0 113.13708 207.87465 16.162441]
/Extend [true true]
/Domain [0 1]
/Function <<
/FunctionType 3
/Functions
[
<<
/FunctionType 2
/Domain [0 1]
/C0 [1 1 1]
/C1 [0.72941178 0.72941178 0.72941178]
/N 1
>>
]
/Domain [0 1]
/Bounds [ ]
/Encode [ 0 1 ]
>>
>>
newpath
5.2721217 202.83181 moveto
5.2804896 203.09247 5.0204563 203.3156 4.7640539 203.26768 curveto

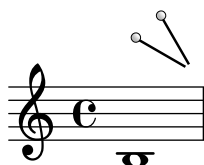
```

```

4.5058701 203.23609 4.3248803 202.94671 4.4100662 202.70074 curveto
4.4779975 202.45848 4.7737511 202.3207 5.0027593 202.42554 curveto
5.1631598 202.49149 5.2743721 202.65813 5.2721217 202.83181 curveto
closepath
clip
gsave [
  0.052859054 0.063089841 -0.020912282 0.017521108 3.363346 191.83136
] concat
shfill
grestore
grestore
0 0 0 setrgbcolor
[] 0 setdash
0.027282091 setlinewidth
0 setlinejoin
0 setlinecap
newpath
5.2721217 202.83181 moveto
5.2804896 203.09247 5.0204563 203.3156 4.7640539 203.26768 curveto
4.5058701 203.23609 4.3248803 202.94671 4.4100662 202.70074 curveto
4.4779975 202.45848 4.7737511 202.3207 5.0027593 202.42554 curveto
5.1631598 202.49149 5.2743721 202.65813 5.2721217 202.83181 curveto
closepath
stroke
grestore
grestore
"
}

\score {
  b1^\stick
}

```



## Percussion example

A short example taken from Stravinsky's *L'histoire du Soldat*.

```

#(define mydrums '((bassdrum  default #f  4)
                    (snare      default #f -4)
                    (tambourine default #f  0)))

```

```

U = \stemUp
D = \stemDown

```

```

global = {
  \time 3/8 s4.
  \time 2/4 s2*2
  \time 3/8 s4.
}

```

```

\time 2/4 s2
}

drumsA = {
  \context DrumVoice <<
    \global
    \drummode {
      \autoBeamOff
      \D sn8 \U tamb s |
      sn4 \D sn4 |
      \U tamb8 \D sn \U sn16 \D sn \U sn8 |
      \D sn8 \U tamb s |
      \U sn4 s8 \U tamb
    }
  >>
}

drumsB = \drummode {
  s4 bd8 s2*2 s4 bd8 s4 bd8 s
}

\layout {
  indent = 40\mm
  \context {
    \DrumStaff
    drumStyleTable = #(alist->hash-table mydrums)
  }
}

\score {
  \new StaffGroup <<
    \new DrumStaff \with {
      instrumentName = \markup \center-column {
        "Tambourine"
        "et"
        "caisse claire s. timbre" }
    } \drumsA
    \new DrumStaff \with {
      instrumentName = "Grosse Caisse"
    } \drumsB
  >>
}

```

Tambourine  
et  
caisse claire s. timbre

Grosse Caisse

## Printing music with different time signatures

In the following snippet, two parts have a completely different time signature, yet remain synchronized.

The bar lines can no longer be printed at the Score level; to allow independent bar lines in each part, the `Default_barline_engraver` and `Timing_translator` are moved from the Score context to the Staff context.

If bar numbers are required, the `Bar_number_engraver` should also be moved, since it relies on properties set by the `Timing_translator`; a `\with` block can be used to add bar numbers to the relevant staff.

```
global = {
  \time 3/4 s2.*3 \break
  s2.*3
}

\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Bar_number_engraver"
    \override SpacingSpanner.uniform-stretching = ##t
    \override SpacingSpanner.strict-note-spacing = ##t
    \proportionalNotationDuration = #1/64
  }
  \context {
    \Staff
    \consists "Timing_translator"
  }
  \context {
    \Voice
    \remove "Forbid_line_break_engraver"
    \tupletFullLength = ##t
  }
}

Bassklarinette = \new Staff \with {
  \consists "Bar_number_engraver"
  \barNumberVisibility = #(every-nth-bar-number-visible 2)
  \override BarNumber.break-visibility = #end-of-line-invisible
} <<
\global
{
  \clef treble
  \time 3/8 d''4. |
  \time 3/4 r8 des''2( c''8) |
  \time 7/8 r4. ees''2 ~ |
  \time 2/4 \tupletUp \tuplet 3/2 { ees''4 r4 d''4 ~ } |
  \time 3/8 \tupletUp \tuplet 4/3 { d''4 r4 } |
  \time 2/4 e''2 |
  \time 3/8 es''4. |
  \time 3/4 r8 d''2 r8 |
}
```

&gt;&gt;

```
Perkussion = \new StaffGroup <<
```

```
  \new Staff <<
```

```
    \global
```

```
    {
```

```
      \clef percussion
```

```
      \time 3/4 r4 c'2 ~ |
```

```
      c'2. |
```

```
      R2. |
```

```
      r2 g'4 ~ |
```

```
      g'2. ~ |
```

```
      g'2. |
```

```
    }
```

&gt;&gt;

```
  \new Staff <<
```

```
    \global {
```

```
      \clef percussion
```

```
      \time 3/4 R2. |
```

```
      g'2. ~ |
```

```
      g'2. |
```

```
      r4 g'2 ~ |
```

```
      g'2 r4 |
```

```
      g'2. |
```

```
    }
```

&gt;&gt;

&gt;&gt;

```
\score {
```

```
  <<
```

```
    \Bassklarinette
```

```
    \Perkussion
```

&gt;&gt;

```
}
```

The image shows a musical score for two parts: Bass Clarinet and Percussion. The Bass Clarinet part is written on a single staff with a treble clef and a 3/4 time signature. It contains a melody with eighth and quarter notes, some beamed together, and a final triplet. The Percussion part consists of two staves, both with a 3/4 time signature. The top staff has a series of eighth notes and quarter notes, while the bottom staff has a series of quarter notes and rests. The two staves of the Percussion part are grouped together with a brace on the left.



(4)

3 4 6

8

## Tam-tam example

A tam-tam example, entered with 'tt'.

```
#(define mydrums '((tamtam default #f 0)))
```

```
\new DrumStaff \with { instrumentName = #"Tamtam" }
```

```
\drummode {
```

```
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
```

```
  \override Staff.StaffSymbol.line-positions = #'( 0 )
```

```
  \override Staff.BarLine.bar-extent = #'(-1.5 . 1.5)
```

```
  tt 1 \pp \laissezVibrer
}
```

Tamtam *pp*

## Tambourine example

A tambourine example, entered with 'tamb'.

```
#(define mydrums '((tambourine default #f 0)))
```

```
\new DrumStaff \with { instrumentName = #"Tambourine" }
```

```
\drummode {
```

```
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
```

```
  \override Staff.StaffSymbol.line-positions = #'( 0 )
```

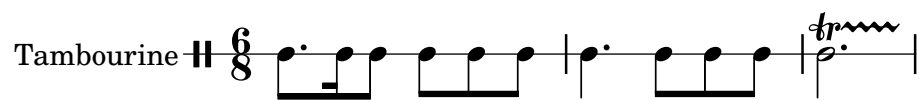
```
  \override Staff.BarLine.bar-extent = #'(-1.5 . 1.5)
```

```
  \time 6/8
```

```

tamb8. 16 8 8 8 8 |
tamb4. 8 8 8 |
% The trick with the scaled duration and the shorter rest
% is neccessary for the correct ending of the trill-span!
tamb2.*5/6 \startTrillSpan s8 \stopTrillSpan |
}

```



## 14 Wind instruments

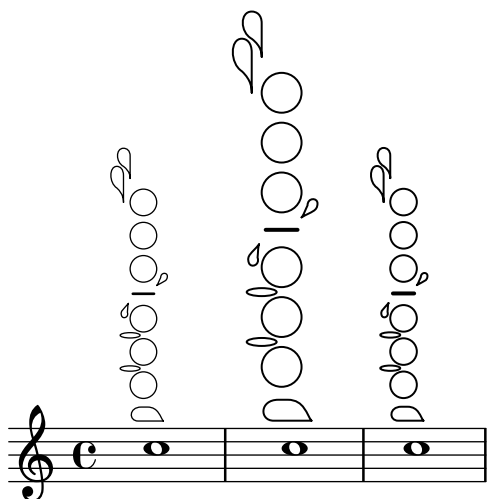
See also Section “Wind instruments” in *Notation Reference*.

### Changing the size of woodwind diagrams

The size and thickness of woodwind diagrams can be changed.

```
\relative c' {
  \textLengthOn
  c1^\markup
    \woodwind-diagram #'piccolo #'()

  c^\markup \override #'(size . 1.5)
    \woodwind-diagram #'piccolo #'()
  c^\markup \override #'(thickness . 0.15)
    \woodwind-diagram #'piccolo #'()
}
```



### Fingering symbols for wind instruments

Special symbols can be achieved by combining existing glyphs, which is useful for wind instruments.

```
lineup =
  \tweak outside-staff-padding #0
  \tweak staff-padding #0
  \tweak padding #0.2
  \tweak parent-alignment-X #CENTER
  \tweak self-alignment-X #CENTER
  \etc

\relative c' {
  g\open
  g\lineup ^\markup \combine
    \musicglyph "scripts.open"
    \musicglyph "scripts.tenuto"
  g\lineup ^\markup \combine
    \musicglyph "scripts.open"
```

```

\musicglyph "scripts.stopped"
g\stopped
}

```



## Flute slap notation

It is possible to indicate special articulation techniques such as a flute “tongue slap” by replacing the note head with the appropriate glyph. For that we can draw the accent-like note head with `\markup`.

```

slap =
#(define-music-function (music) (ly:music?)
  #{
    \temporary \override NoteHead.stencil =
      #ly:text-interface::print
    \temporary \override NoteHead.text =
      \markup
        \translate #'(1 . 0)
        \override #'(thickness . 1.4)
        \overlay { \draw-line #'(-1.2 . 0.4)
                  \draw-line #'(-1.2 . -0.4) }
    \temporary \override NoteHead.stem-attachment =
      #(lambda (grob)
        (let* ((stem (ly:grob-object grob 'stem))
              (dir (ly:grob-property stem 'direction UP))
              (is-up (eqv? dir UP)))
          (cons dir (if is-up 0 -0.8)))))
    #music
    \revert NoteHead.stencil
    \revert NoteHead.text
    \revert NoteHead.stem-attachment
  })

\relative c' {
  c4 \slap c d r
  \slap { g4 a } b r
}

```



## Graphical and text woodwind diagrams

In many cases, the keys other than the central column can be displayed by key name as well as by graphical means.

```

\relative c' {
  \textLengthOn
  c1^\markup

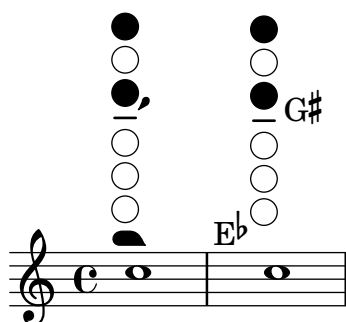
```

```

\woodwind-diagram #'piccolo
      #'((cc . (one three))
        (lh . (gis))
        (rh . (ees)))

c^\markup
  \override #'(graphical . #f)
  \woodwind-diagram #'piccolo
      #'((cc . (one three))
        (lh . (gis))
        (rh . (ees)))
}

```



## Recorder fingering chart

The following example demonstrates how fingering charts for wind instruments can be realized.

*% range chart for paetzold contrabass recorder*

```

centermarkup = {
  \once \override TextScript.self-alignment-X = #CENTER
  \once \override TextScript.X-offset = #(lambda (g)
    (+ (ly:self-alignment-interface::centered-on-x-parent g)
      (ly:self-alignment-interface::x-aligned-on-self g)))
}

\new Staff \with {
  \remove "Time_signature_engraver"
  \omit Stem
  \omit Flag
  \consists "Horizontal_bracket_engraver"
} {
  \clef bass
  \set Score.timing = ##f

  f,1*1/4 \glissando

  \clef violin
  gis'1*1/4

  a'4^\markup "1)"

  \centermarkup
}

```

```

\once \override TextScript.padding = 2
bes'1*1/4_\markup \override #'(baseline-skip . 1.7) \column {
  \fontsize #-5
  \slashed-digit #0 \finger 1 \finger 2
  \finger 3 \finger 4 \finger 5 \finger 6 \finger 7 }

b'1*1/4

c''4^\markup "1)"

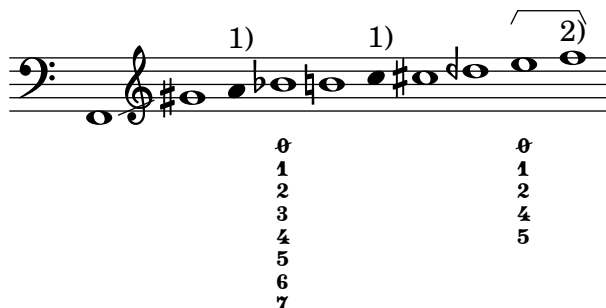
cis''1*1/4

deh''1*1/4

\centermarkup
\once \override TextScript.padding = 2
\once \override Staff.HorizontalBracket.direction = #UP
e''1*1/4_\markup \override #'(baseline-skip . 1.7) \column {
  \fontsize #-5
  \slashed-digit #0 \finger 1 \finger 2
  \finger 4 \finger 5 } \startGroup

f''1*1/4^\markup "2)" \stopGroup
}

```



## 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 gets written to `stderr` but is not shown in the music. If output to `stdout` is wanted instead, omit the code `(current-error-port)` from the commands.

```

#(print-keys-verbose 'piccolo (current-error-port))
#(print-keys-verbose 'flute (current-error-port))
#(print-keys-verbose 'flute-b-extension (current-error-port))
#(print-keys-verbose 'tin-whistle (current-error-port))
#(print-keys-verbose 'oboe (current-error-port))
#(print-keys-verbose 'clarinet (current-error-port))
#(print-keys-verbose 'bass-clarinet (current-error-port))
#(print-keys-verbose 'low-bass-clarinet (current-error-port))
#(print-keys-verbose 'saxophone (current-error-port))
#(print-keys-verbose 'soprano-saxophone (current-error-port))
#(print-keys-verbose 'alto-saxophone (current-error-port))
#(print-keys-verbose 'tenor-saxophone (current-error-port))

```

```

#(print-keys-verbose 'baritone-saxophone (current-error-port))
#(print-keys-verbose 'bassoon (current-error-port))
#(print-keys-verbose 'contrabassoon (current-error-port))

\score {c''1}

```



## Woodwind diagrams listing

The following music shows all of the woodwind diagrams currently defined in LilyPond.

```

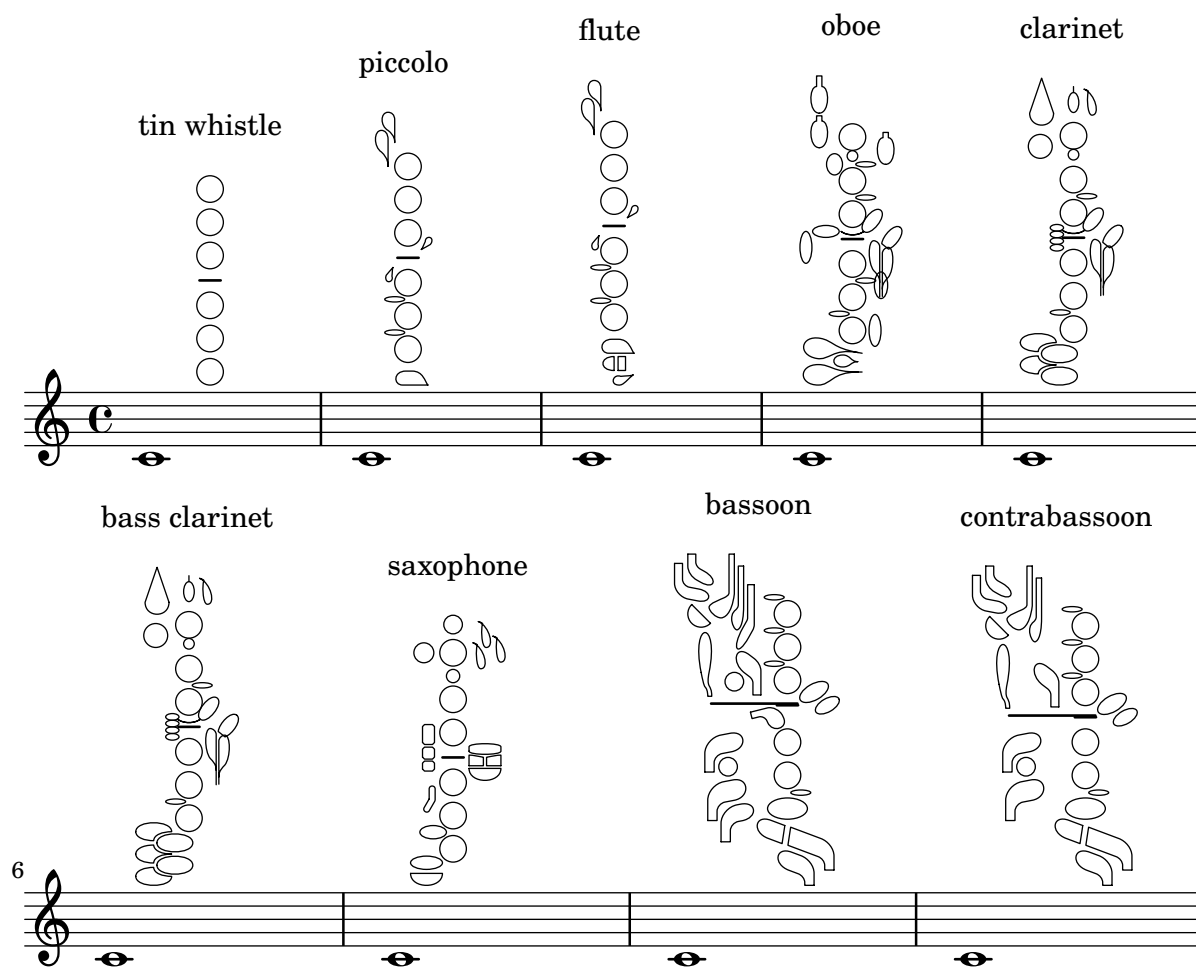
\relative c' {
  \textLengthOn
  c1^\markup \center-column { "tin whistle"
    " "
    \woodwind-diagram #'tin-whistle #'() }
  c1^\markup \center-column { "piccolo"
    " "
    \woodwind-diagram #'piccolo #'() }
  c1^\markup \center-column { "flute"
    " "
    \woodwind-diagram #'flute #'() }
  c1^\markup \center-column { "oboe"
    " "
    \woodwind-diagram #'oboe #'() }
  c1^\markup \center-column { "clarinet"
    " "
    \woodwind-diagram #'clarinet #'() }

  \break

  c1^\markup \center-column { "bass clarinet"
    " "
    \woodwind-diagram #'bass-clarinet #'() }
  c1^\markup \center-column { "saxophone"
    " "
    \woodwind-diagram #'saxophone #'() }
  c1^\markup \center-column { "bassoon"
    " "
    \woodwind-diagram #'bassoon #'() }
  c1^\markup \center-column { "contrabassoon"
    " "
    \woodwind-diagram #'contrabassoon #'() }
}

\paper {
  system-system-spacing.padding = 5
}

```





## 15 Chord notation

See also Section “Chord notation” in *Notation Reference*.

### Adding a figured bass above or below the notes

When writing figured bass, you can place the figures above or below the bass notes by using the commands `\bassFigureStaffAlignmentDown` and `\bassFigureStaffAlignmentUp`. Prepend `\once` to the command if you want to modify only the next figured bass.

The command `\bassFigureStaffAlignmentNeutral` resets the direction of figured bass to the default value.

```
bass = {
  \clef bass
  g4 b, c d |
  e d8 c d2
}

continuo = \figuremode {
  <_>4 <6>4 <5/>4
  \bassFigureStaffAlignmentUp
  <_+>4 <6> |
  \set Staff.useBassFigureExtenders = ##t
  \bassFigureStaffAlignmentDown
  <4>4. <4>8 <_+>4
}

\score {
  <<
    \new Staff = bassStaff \bass
    \context Staff = bassStaff \continuo
  >>
}
```



### 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 = #'(-1 . 3)
  \consists "Bar_engraver"
}

\chordmode {
  f1:maj7 f:7 bes:7
}
```

F<sup>Δ</sup> | F<sup>7</sup> | B<sup>b</sup>7 |

## Adjusting figured bass alteration glyphs

In figured bass, specially designed glyphs for 6\\, 7\\, and 9\\ are used by default. Similarly, specially designed glyphs for symbols 2\\+, 4\\+, and 5\\+ are used by default if plus signs appear after the number.

To change that, pass an alist to `figuredBassPlusStrokedAlist` and set the glyph in question to #f (or omit it).

```
#(set-global-staff-size 26)
```

```
\figures {
  \set figuredBassPlusDirection = #RIGHT
  <6\\> <7\\> <9\\> r
  <2\\+> <4\\+> <5\\+> r

  \set figuredBassPlusStrokedAlist =
    #'((2 . "figbass.twoplus")
      ;; (4 . "figbass.fourplus")
      ;; (5 . "figbass.fiveplus")
      (6 . "figbass.sixstroked")
      ;; (7 . "figbass.sevenstroked")
      ;; (9 . "figbass.ninestroked")
    )
  <6\\> <7\\> <9\\> r
  <2\\+> <4\\+> <5\\+> r
}
```

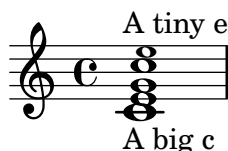
6 7 9                      2 4 5+                      6 7 9                      2 4+ 5+

## Changing a single note's size in a chord

Individual note heads in a chord can be modified with the `\tweak` command inside a chord, by altering the font-size property.

Inside the chord (within the brackets < >), before the note to be altered, place the `\tweak` command, followed by font-size and define the proper size like # -2 (a tiny note head).

```
\relative c' {
  <\tweak font-size #+2 c e g c
  \tweak font-size #-2 e>1
  ~\markup { A tiny e }_~\markup { A big c }
}
```



## Changing chord separator

The separator between different parts of a chord name can be set to any markup.

```
\chords {
  c:7sus4
  \set chordNameSeparator = \markup { \typewriter | }
  c:7sus4
}
```

}

$C^7 \text{ sus4 } C^7 | \text{ sus4}$

## 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.

If plus signs appear after the number, specially designed glyphs are provided for some figures.  `#(set-global-staff-size 26)`

```
\figures {
  <5\+> <5+ 4\+> <6 4- 2\+> r
  \set figuredBassAlterationDirection = #RIGHT
  <5\+> <5+ 4\+> <6 4- 2\+> r
  \set figuredBassPlusDirection = #RIGHT
  <5\+> <5+ 4\+> <6 4- 2\+> r
  \set figuredBassAlterationDirection = #LEFT
  <5\+> <5+ 4\+> <6 4- 2\+> r
}
```

$+5 \quad \sharp 5 \quad 6$ 
 $+5 \quad 5\sharp \quad 6$ 
 $5^+ \quad 5\sharp \quad 6$ 
 $5^+ \quad \sharp 5 \quad 6$

$+4 \quad \flat 4 \quad +2$ 
 $+4 \quad 4\flat \quad +2$ 
 $4^+ \quad 4\flat \quad 2^+$ 
 $4^+ \quad \flat 4 \quad 2^+$

## Chord name exceptions

The property `chordNameExceptions` stores a list of chord name exceptions to handle cases either not covered or handled incorrectly.

The default chord names used by LilyPond follow the rules as given in Klaus Ignatzek's book "Die Jazzmethode für Klavier 1"; the algorithm to convert chords to chord names can be found in file `scm/chord-ignatzek-names.scm`. Additional rules are given as chord exceptions and stored in the variable `ignatzekExceptions`, as set up in file `ly/chord-modifiers-init.ly`.

This snippet modifies these exceptions in three steps.

1. Set up some music with chords and associated markup. By convention, the root (i.e., the lowest note) of each chord should have pitch `c`.
2. Call Scheme function `sequential-music-to-chord-exceptions` to create a new list of exceptions, then concatenate it with the existing ones. Since `ignatzekExceptions` is set up with this function's second parameter set to `#t` (to ignore the root of the chords), we have to do the same.
3. Register the new exception list.

*% Step 1: Define music with chords and markup for maj9 and 6(add9).*

```
chExceptionMusic = {
  <c e g b d'>-\markup { \super "maj9" }
  <c e g a d'>-\markup { \super "6(add9)" }
}
```

*% Step 2: Create extended exception list.*

```
chExceptions =
#(append (sequential-music-to-chord-exceptions chExceptionMusic #t)
  ignatzekExceptions)
```

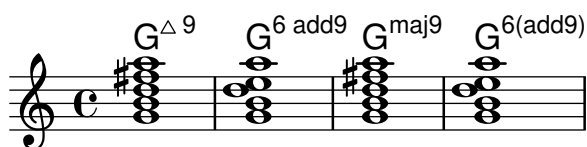
```

theMusic = \chordmode {
  g1:maj9 g1:6.9
  % Step 3: Register extended exception list.
  \set chordNameExceptions = #chExceptions
  g1:maj9 g1:6.9
}

\layout {
  ragged-right = ##t
}

<<
  \new ChordNames \theMusic
  \new Voice \theMusic
>>

```



## Chord name major7

The layout of the major 7 can be tuned with `majorSevenSymbol`.

```

\chords {
  c:7+
  \set majorSevenSymbol = \markup { j7 }
  c:7+
}

```

$C^{\Delta} C^{j7}$

## Chord names alternative

Chord names are generated from a list of pitches. The functions which construct these names can be customised.

Here are shown chords following Ignatzek (pp. 17-18, 1995), used by default since LilyPond 1.7.20, compared with an alternative Jazz chord notation and Harald Banter's (1987) notation. A smaller font is used in the latter case, as these tend to be overly verbose.

This mirrors the mechanism originally used in early LilyPond versions (pre-1.7); not having been properly maintained, however, some features have been lost (mainly chord exception lists) and bugs have been introduced.

```

%%% Legacy chord naming functions (formerly in scm/chord-generic-names.scm)
%%% Copyright (C) 2003--2023 Jan Nieuwenhuizen <janneke@gnu.org>

```

```

#(set-global-staff-size 19.7)

#(define-public (banter-chordnames pitches bass inversion context)
  (old_chord->markup 'banter pitches bass inversion context))

#(define-public (jazz-chordnames pitches bass inversion context)
  (old_chord->markup 'jazz pitches bass inversion context))

```

```

#(define (define-translator-property symbol type? description)
  (if (not (and (symbol? symbol)
                (procedure? type?)
                (string? description)))
      (ly:error "error in call of define-translator-property"))
  (if (not (equal? (object-property symbol 'translation-doc) #f))
      (ly:error (G_ "symbol ~S redefined") symbol))

  (set-object-property! symbol 'translation-type? type?)
  (set-object-property! symbol 'translation-doc description)
  symbol)

#(for-each
  (lambda (x)
    (apply define-translator-property x))
  `((chordNameExceptionsFull ,list? "An alist of full chord
exceptions. Contains @code{(@var{chord} . @var{markup})} entries.")
    (chordNameExceptionsPartial ,list? "An alist of partial chord
exceptions. Contains @code{(@var{chord} . (@var{prefix-markup}
@var{suffix-markup}))} entries.")))

#(define-public (old_chord->markup
                 style pitches bass inversion context)
  "Entry point for @code{Chord_name_engraver}.
@var{pitches}, @var{bass}, and @var{inversion} are lily pitches."
  (define (default-note-namer pitch)
    (note-name->markup pitch #f))

  (define (markup-or-empty-markup markup)
    "Return MARKUP if markup, else empty-markup"
    (if (markup? markup) markup empty-markup))

  (define (accidental->markup alteration)
    "Return accidental markup for ALTERATION."
    (if (= alteration 0)
        (make-line-markup (list empty-markup))
        (conditional-kern-before
         (alteration->text-accidental-markup alteration)
         (= alteration FLAT) 0.094725))))

(define (list-minus a b)
  "Return list of elements in A that are not in B."
  (lset-difference eq? a b))

(define (markup-join markups sep)
  "Return line-markup of MARKUPS, joining them with markup SEP"
  (if (pair? markups)
      (make-line-markup (list-insert-separator markups sep))
      empty-markup))

(define (conditional-kern-before markup bool amount)

```

```

"Add AMOUNT of space before MARKUP if BOOL is true."
(if bool
  (make-line-markup
    (list (make-hspace-markup amount)
          markup))
  markup))

(define (step-nr pitch)
  (let* ((pitch-nr (+ (* 7 (ly:pitch-octave pitch))
                     (ly:pitch-notename pitch)))
        (root-nr (+ (* 7 (ly:pitch-octave (car pitches))
                     (ly:pitch-notename (car pitches))))
        (+ 1 (- pitch-nr root-nr))))

(define (next-third pitch)
  (+ pitch
     (ly:make-pitch 0 2 (if (or (= (step-nr pitch) 3)
                                (= (step-nr pitch) 5))
                           FLAT 0))))

(define (step-alteration pitch)
  (let* ((normalized-pitch (- pitch (car pitches)))
        (alteration (ly:pitch-alteration normalized-pitch)))
    (if (= (step-nr pitch) 7) (+ alteration SEMI-TONE) alteration)))

(define (pitch-unalter pitch)
  (let ((alteration (step-alteration pitch)))
    (if (= alteration 0)
        pitch
        (ly:make-pitch (ly:pitch-octave pitch) (ly:pitch-notename pitch)
                        (- (ly:pitch-alteration pitch) alteration)))))

(define (step-even-or-altered? pitch)
  (let ((nr (step-nr pitch)))
    (if (!= (modulo nr 2) 0)
        (!= (step-alteration pitch) 0)
        #t)))

(define (step->markup-plusminus pitch)
  (let ((alt (step-alteration pitch)))
    (make-line-markup
     (list
      (number->string (step-nr pitch))
      (cond
       ((= alt DOUBLE-FLAT) "--")
       ((= alt FLAT) "-")
       ((= alt NATURAL) "")
       ((= alt SHARP) "+")
       ((= alt DOUBLE-SHARP) "++"))))))

(define (step->markup-accidental pitch)
  (make-line-markup

```

```

      (list (accidental->markup (step-alteration pitch))
            (make-simple-markup (number->string (step-nr pitch))))))

(define (step->markup-ignatzek pitch)
  (make-line-markup
   (if (and (= (step-nr pitch) 7)
             (= (step-alteration pitch) 1))
       (list (ly:context-property context 'majorSevenSymbol))
       (list (accidental->markup (step-alteration pitch))
             (make-simple-markup (number->string (step-nr pitch))))))

;; tja, kennok
(define (make-sub->markup step->markup)
  (lambda (pitch)
    (make-line-markup (list (make-simple-markup "no")
                           (step->markup pitch)))))

(define (step-based-sub->markup step->markup pitch)
  (make-line-markup (list (make-simple-markup "no") (step->markup pitch))))

(define (get-full-list pitch)
  (if (<= (step-nr pitch) (step-nr (last pitches)))
      (cons pitch (get-full-list (next-third pitch)))
      '()))

(define (get-consecutive nr pitches)
  (if (pair? pitches)
      (let* ((pitch-nr (step-nr (car pitches)))
             (next-nr (if (!= (modulo pitch-nr 2) 0) (+ pitch-nr 2) nr)))
        (if (<= pitch-nr nr)
            (cons (car pitches) (get-consecutive next-nr (cdr pitches)))
            '()))
      '()))

;;; FIXME -- exceptions no longer work. -vv

(define (full-match exceptions)
  (if (pair? exceptions)
      (let* ((e (car exceptions))
             (e-pitches (car e)))
        (if (equal? e-pitches pitches)
            e
            (full-match (cdr exceptions)))))
  #f))

(define (partial-match exceptions)
  (if (pair? exceptions)
      (let* ((e (car exceptions))
             (e-pitches (car e)))
        (if (equal? e-pitches (take pitches (length e-pitches)))
            e
            (partial-match (cdr exceptions)))))
  #f))

```

```

#f))

;; FIXME: exceptions don't work anyway.
(if #f (begin
  (write-me "pitches: " pitches)))
(let* ((full-exceptions
  (ly:context-property context 'chordNameExceptionsFull))
  (full-exception (full-match full-exceptions))
  (full-markup (if full-exception (cadr full-exception) '()))
  (partial-exceptions
  (ly:context-property context 'chordNameExceptionsPartial))
  (partial-exception (partial-match partial-exceptions))
  (partial-pitches (if partial-exception (car partial-exception) '()))
  (partial-markup-prefix
  (if partial-exception (markup-or-empty-markup
    (cadr partial-exception)) empty-markup))
  (partial-markup-suffix
  (if (and partial-exception (pair? (cddr partial-exception)))
    (markup-or-empty-markup (caddr partial-exception)) empty-markup))
  (root (car pitches))
  (full (get-full-list root))
  ;; kludge alert: replace partial matched lower part of all with
  ;; 'normal' pitches from full
  ;; (all pitches)
  (all (append (take full (length partial-pitches))
    (drop pitches (length partial-pitches))))

  (highest (last all))
  (missing (list-minus full (map pitch-unalter all)))
  (consecutive (get-consecutive 1 all))
  (rest (list-minus all consecutive))
  (altered (filter step-even-or-altered? all))
  (cons-alt (filter step-even-or-altered? consecutive))
  (base (list-minus consecutive altered)))

  (if #f (begin
    (write-me "full:" full)
    ;; (write-me "partial-pitches:" partial-pitches)
    (write-me "full-markup:" full-markup)
    (write-me "partial-markup-perfix:" partial-markup-prefix)
    (write-me "partial-markup-suffix:" partial-markup-suffix)
    (write-me "all:" all)
    (write-me "altered:" altered)
    (write-me "missing:" missing)
    (write-me "consecutive:" consecutive)
    (write-me "rest:" rest)
    (write-me "base:" base)))

  (case style
    ((banter)
     ;; root

```



```

;; + steps:altered + (highest all -- if not altered)
;; + subs:missing

(let* ((root->markup default-note-namer)
      (step->markup step->markup-plusminus)
      (sub->markup (lambda (x)
                     (step-based-sub->markup step->markup x)))
      (sep (make-simple-markup "/")))

  (if
    (pair? full-markup)
    (make-line-markup (list (root->markup root) full-markup))

    (make-line-markup
      (list
        (root->markup root)
        partial-markup-prefix
        (make-super-markup
          (markup-join
            (append
              (map step->markup
                (append altered
                  (if (and (> (step-nr highest) 5)
                        (not
                          (step-even-or-altered? highest)))
                    (list highest) '()))))
              (list partial-markup-suffix)
              (map sub->markup missing))
            sep))))))

((jazz)
  ;; root
  ;; + steps:(highest base) + cons-alt
  ;; + 'add'
  ;; + steps:rest
  (let* ((root->markup default-note-namer)
        (step->markup step->markup-ignatzek)
        (sep (make-simple-markup " "))
        (add-prefix (make-simple-markup " add")))

    (if
      (pair? full-markup)
      (make-line-markup (list (root->markup root) full-markup))

      (make-line-markup
        (list
          (root->markup root)
          partial-markup-prefix
          (make-super-markup
            (make-line-markup
              (list

```

```

;; kludge alert: omit <= 5
;;(markup-join (map step->markup
;;              (cons (last base) cons-alt)) sep)

;; This fixes:
;; c      C5      -> C
;; c:2    C5 2    -> C2
;; c:3-   Cm5     -> Cm
;; c:6.9 C5 6add9 -> C6 add 9 (add?)
;; ch = \chords { c c:2 c:3- c:6.9~7 }
(markup-join (map step->markup
                  (let ((tb (last base)))
                    (if (> (step-nr tb) 5)
                        (cons tb cons-alt)
                        cons-alt)))) sep)

(if (pair? rest)
    add-prefix
    empty-markup)
(markup-join (map step->markup rest) sep)
partial-markup-suffix))))))

(else empty-markup)))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% Here begins the actual snippet:

chs = \transpose c' c' {
  <c e g>1
  <c es g> % m = minor triad
  <c e gis>
  <c es ges> \break
  <c e g bes>
  <c es g bes>
  <c e g b> % triangle = maj
  <c es ges beses>
  <c es ges b> \break
  <c e gis bes>
  <c es g b>
  <c e gis b>
  <c es ges bes> \break
  <c e g a> % 6 = major triad with added sixth
  <c es g a> % m6 = minor triad with added sixth
  <c e g bes d'>
  <c es g bes d'> \break
  <c es g bes d' f' a' >
  <c es g bes d' f' >
  <c es ges bes d' >
  <c e g bes des' > \break

```

```

<c e g bes dis'>
<c e g bes d' f'>
<c e g bes d' fis'>
<c e g bes d' f' a'> \break
<c e g bes d' fis' as'>
<c e gis bes dis'>
<c e g bes dis' fis'>
<c e g bes d' f' as'> \break
<c e g bes des' f' as'>
<c e g bes d' fis'>
<c e g b d'>
<c e g bes d' f' as'> \break
<c e g bes des' f' as'>
<c e g bes des' f' a'>
<c e g b d'>
<c e g b d' f' a'> \break
<c e g b d' fis'>
<c e g bes des' f ' a'>
<c f g>
<c f g bes> \break
<c f g bes d'>
<c e g d'> % add9
<c es g f'>
<c e g b fis'> % Lydian
<c e g bes des' ees' fis' aes'> % altered chord
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% alternate Jazz notation

efullmusicJazzAlt = {
  <c e gis>1-\markup { "+" }
  <c e g b>-\markup {
    \normal-size-super
    % \override #'(font-family . math) "N"
    \override #'(font-family . math) "M"
  }
  %%c:3.5.7 = \markup { \override #'(font-family . math) "M" }
  %%c:3.5.7 = \markup { \normal-size-super "maj7" }

  <c es ges>-\markup { \super "o" } % should be $\circ$ ?
  <c es ges bes>-\markup { \super \combine "o" "/" }
  <c es ges beses>-\markup { \super "o7" }
}

efullJazzAlt = #(sequential-music-to-chord-exceptions efullmusicJazzAlt #f)

epartialmusicJazzAlt = {
  <c d>1-\markup { \normal-size-super "2" }
  <c es>-\markup { "m" }
  <c f>-\markup { \normal-size-super "sus4" }
}

```

```

<c g>-\markup { \normal-size-super "5" }
%% TODO, partial exceptions
<c es f>-\markup { "m" }-\markup { \normal-size-super "sus4" }
<c d es>-\markup { "m" }-\markup { \normal-size-super "sus2" }
}

epartialJazzAlt = #(sequential-music-to-chord-exceptions epartialmusicJazzAlt #f)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\score {
  <<
    \new ChordNames {
      %% Already set by default:
      \%set chordNameFunction = #ignatzek-chord-names
      \set instrumentName = "Ignatzek"
      \set shortInstrumentName = "Def"
      \chs
    }

    \new ChordNames {
      \set chordNameFunction = #jazz-chordnames
      \set majorSevenSymbol = \whiteTriangleMarkup
      \set chordNameSeparator = "/"
      \set chordNameExceptionsFull = \efullJazzAlt
      \set chordNameExceptionsPartial = \epartialJazzAlt
      \set instrumentName = "Alternative"
      \set shortInstrumentName = "Alt"
      \chs
    }

    %% This is the Banter (1987) style.  It gives exceedingly
    %% verbose (wide) names, making the output file take up to 4 pages.

    \new ChordNames {
      \set chordNameFunction = #banter-chordnames
      \override ChordName.font-size = -3
      \set instrumentName = "Banter"
      \set shortInstrumentName = "Ban"
      \chs
    }


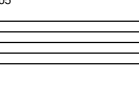

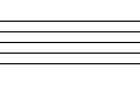

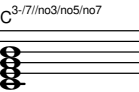
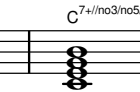
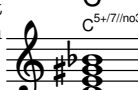
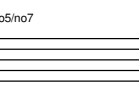
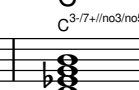
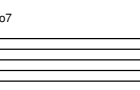

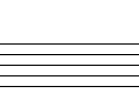
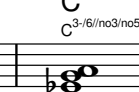
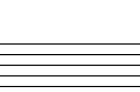
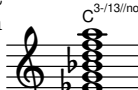
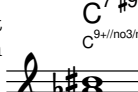
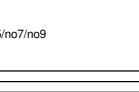
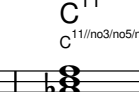
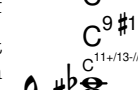
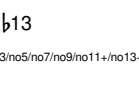
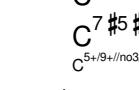

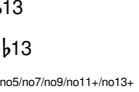
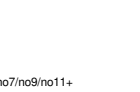
  \new Staff \transpose c c' { \chs }
  >>
  \layout {
    #(layout-set-staff-size 16)
    system-system-spacing.basic-distance = 0
    \context {
      \ChordNames
      \consists "Instrument_name_engraver"
    }
    \context {

```

```

\Score
\remove "Bar_number_engraver"
}
}
}

```

Ignatzek	C	Cm	C+	C°
Alternative	C	C <sub>b3</sub>	C <sub>#5</sub>	C <sub>b3 b5</sub>
Banter	C <sub>no3/no5</sub>	C <sub>3-//no3/no5</sub>	C <sub>5+//no3/no5</sub>	C <sub>3-5-//no3/no5</sub>
				
Def	C <sup>7</sup>	Cm <sup>7</sup>	C <sup>Δ</sup>	C <sup>o7</sup>
Alt	C <sup>7</sup>	C <sup>7</sup> <sub>b3</sub>	C <sub>#7</sub>	C <sub>b3 b5 b7</sub>
Ban	C <sub>7//no3/no5/no7</sub>	C <sub>3-7//no3/no5/no7</sub>	C <sub>7+//no3/no5/no7</sub>	C <sub>3-5-7-//no3/no5/no7</sub>
				
Def	C <sup>7</sup> <sub>#5</sub>	Cm <sup>Δ</sup>	C <sup>Δ</sup> <sub>#5</sub>	C <sup>ø</sup>
Alt	C <sup>7</sup> <sub>#5</sub>	C <sub>b3 #7</sub>	C <sub>#5 #7</sub>	C <sub>7 b3 b5</sub>
Ban	C <sub>5+7//no3/no5/no7</sub>	C <sub>3-7+//no3/no5/no7</sub>	C <sub>5+7+//no3/no5/no7</sub>	C <sub>3-5-7//no3/no5/no7</sub>
				
Def	C <sup>6</sup>	Cm <sup>6</sup>	C <sup>9</sup>	Cm <sup>9</sup>
Alt	C <sup>6</sup>	C <sub>b3 6</sub>	C <sup>9</sup>	C <sub>9 b3</sub>
Ban	C <sub>6//no3/no5</sub>	C <sub>3-6//no3/no5</sub>	C <sub>9//no3/no5/no7/no9</sub>	C <sub>3-9//no3/no5/no7/no9</sub>
				
Def	Cm <sup>13</sup>	Cm <sup>11</sup>	Cm <sup>7 b5 9</sup>	C <sup>7 b9</sup>
Alt	C <sub>13 b3</sub>	C <sub>11 b3</sub>	C <sub>9 b3 b5</sub>	C <sub>7 b9</sub>
Ban	C <sub>3-13//no3/no5/no7/no9/no11+/no13+</sub>	C <sub>3-11//no3/no5/no7/no9/no11+</sub>	C <sub>3-5-9//no3/no5/no7/no9</sub>	C <sub>9-//no3/no5/no7/no9</sub>
				
Def	C <sup>7</sup> <sub>#9</sub>	C <sup>11</sup>	C <sup>7</sup> <sub>#11</sub>	C <sup>13</sup>
Alt	C <sup>7</sup> <sub>#9</sub>	C <sup>11</sup>	C <sub>9 #11</sub>	C <sup>13</sup>
Ban	C <sub>9+//no3/no5/no7/no9</sub>	C <sub>11//no3/no5/no7/no9/no11+</sub>	C <sub>11+//no3/no5/no7/no9/no11+</sub>	C <sub>13//no3/no5/no7/no9/no11+/no13+</sub>
				
Def	C <sup>7</sup> <sub>#11 b13</sub>	C <sup>7</sup> <sub>#5 #9</sub>	C <sup>7</sup> <sub>#9 #11</sub>	C <sup>7</sup> <sub>b13</sub>
Alt	C <sub>9 #11 b13</sub>	C <sub>7 #5 #9</sub>	C <sub>7 #9 #11</sub>	C <sub>11 b13</sub>
Ban	C <sub>11+13-//no3/no5/no7/no9/no11+/no13+</sub>	C <sub>5+9+//no3/no5/no7/no9</sub>	C <sub>9+11+//no3/no5/no7/no9/no11+</sub>	C <sub>13-//no3/no5/no7/no9/no11+/no13+</sub>
				
Def	C <sup>7</sup> <sub>b9 b13</sub>	C <sup>7</sup> <sub>#11</sub>	C <sup>Δ</sup> <sub>9</sub>	C <sup>7</sup> <sub>b13</sub>
Alt	C <sub>11 b9 b13</sub>	C <sub>9 #11</sub>	C <sub>9 #7</sub>	C <sub>11 b13</sub>
Ban	C <sub>9-13-//no3/no5/no7/no9/no11+/no13+</sub>	C <sub>11+//no3/no5/no7/no9/no11+</sub>	C <sub>7+9//no3/no5/no7/no9</sub>	C <sub>13-//no3/no5/no7/no9/no11+/no13+</sub>
				

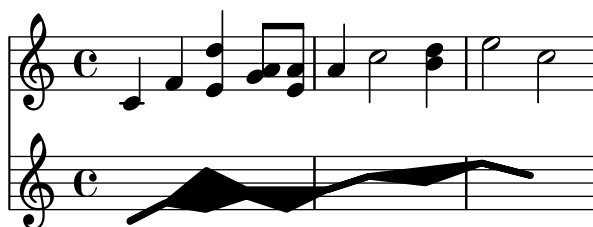


## Clusters

Clusters are a device to denote that a complete range of notes is to be played.

```
fragment = \relative c' {
  c4 f <e d'>4
  <g a>8 <e a> a4 c2 <d b>4
  e2 c
}

<<
  \new Staff \fragment
  \new Staff \makeClusters \fragment
>>
```

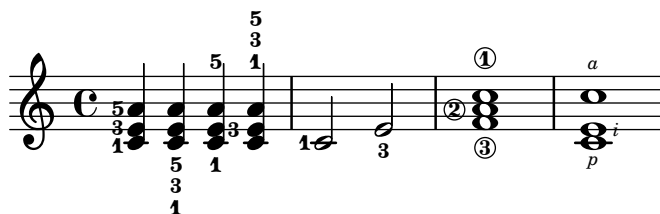


## Controlling the placement of chord fingerings

The placement of fingering numbers can be controlled precisely by using the property `fingeringOrientation`. For fingering orientation to apply, the fingering command must be used within a chord construct (`<...>`), even for single notes. Orientation for string numbers and right-hand fingerings may be controlled in a similar way by using the properties `stringNumberOrientation` and `strokeFingerOrientation`, respectively.

These properties can be set to a list of one to three values. They control whether fingerings may be placed above (if `up` appears in the list), below (if `down` appears), to the left (if `left` appears), or to the right (if `right` appears). Conversely, if a location is not listed, no fingering is placed there. LilyPond takes these constraints and works out the best placement for the fingering of the notes of the following chords. Note that `left` and `right` are mutually exclusive – fingerings may be placed only on one side or the other, not both.

```
\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
  \set stringNumberOrientations = #'(up left down)
  <f\3 a\2 c\1>1
  \set strokeFingerOrientations = #'(down right up)
  <c\rightHandFinger 1 e\rightHandFinger 2 c'\rightHandFinger 4 >
}
```



## Cross-staff chords – beaming problems workaround

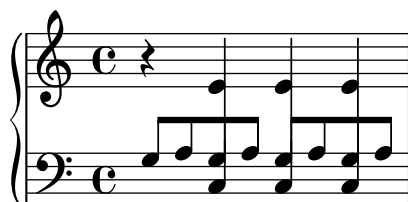
Sometimes it is better to use stems from the ‘other’ staff for creating cross-staff chords to trick LilyPond’s beam collision detector. In the following snippet, if the stems from the lower staff were used instead, it would be necessary to explicitly use

```
\override Staff.Beam.collision-voice-only = ##t
```

so that LilyPond doesn’t move the beams.

```
\new PianoStaff <<
  \new Staff = up \relative c' <<
    { r4
      \override Stem.cross-staff = ##t
      \override Stem.length = #19 % this is in half-spaces,
        % so it makes stems 9.5 staffspaces long
      \override Stem.Y-offset = #-6 % stems are normally lengthened
        % upwards, so here we must lower the stem by the amount
        % equal to the lengthening - in this case (19 - 7) / 2
        % (7 is default stem length)
      e e e }
    { s4
      \change Staff = "bottom"
      \override NoteColumn.ignore-collision = ##t
      c, c c
    }
  >>

  \new Staff = bottom \relative c' {
    \clef bass
    \voiceOne
    g8 a g a g a g a
  }
>>
```



## Customizing the chord grid style

Custom divisions of chord squares can be defined through the `measure-division-lines-alist` and `measure-division-chord-placement-alist` properties of `ChordSquare`. These are both alists. Their keys are measure divisions, namely lists which give the fraction of the measure that each chord (or rest, or skip) represents. More precisely, a measure division alist is made of posi-



tive, exact numbers adding up to 1, for example: '(1/2 1/4 1/4). The exactness requirement means that, e.g., 1/2 is valid but not 0.5.

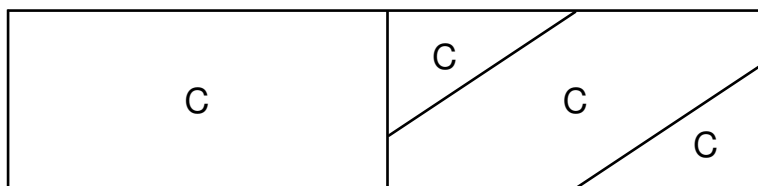
The values in `measure-division-lines-alist` are lists of lines, which are represented as  $(x1\ y1\ x2\ y2)$ . The line starts at the point  $(x1\ .\ y1)$  and ends at  $(x2\ .\ y2)$ . Coordinates are expressed in the  $[-1, 1]$  scale relative to the extent of the square.

The values in `measure-division-chord-placement-alist` are lists of  $(x\ .\ y)$  pairs giving the placement of the respective chords.

This example defines a peculiar chord grid style that has a rule for measures divided in three equal parts.

```
\paper {
  line-width = 10\cm
  ragged-right = ##f
}

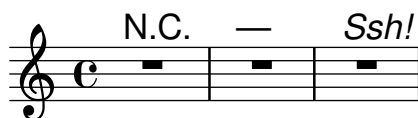
\new ChordGrid \with {
  \override ChordSquare.measure-division-lines-alist =
    #'(((1) . (0))
      ((1/3 1/3 1/3) . ((-1 -0.4 0 1) (0 -1 1 0.4))))
  \override ChordSquare.measure-division-chord-placement-alist =
    #'(((1) . ((0 . 0)))
      ((1/3 1/3 1/3) . ((-0.7 . 0.5) (0 . 0) (0.7 . -0.5))))
}
\chordmode {
  \time 3/4
  c2.
  c4 c4 c4
}
```



## Customizing the no-chord symbol

By default, rests in a `ChordNames` context cause the text “N.C.” to be printed. This markup can be customized by setting the `noChordSymbol` context property.

```
<<
\chords {
  R1
  \set noChordSymbol = "----"
  R1
  \set noChordSymbol = \markup \italic "Ssh!"
  R1
}
{
  R1*3
}
>>
```



## Display non-English chord names

The default English naming of chords can be changed to other languages, as demonstrated in this snippet.

```
scm = \chordmode {
  c1/c | cis/cis
  b1/b | bis/bis | bes/bes
}

\layout {
  indent = 3\cm
  ragged-right = ##f

  \context {
    \ChordNames
    \consists "Instrument_name_engraver"
  }
  \context {
    \Score
    \override InstrumentName.self-alignment-Y = -1.2
    \override InstrumentName.self-alignment-X = #RIGHT
  }
}

<<
\new ChordNames {
  \set instrumentName = #"default"
  \scm
}
\new ChordNames {
  \set instrumentName = #"german"
  \germanChords \scm
}
\new ChordNames {
  \set instrumentName = #"semi-german"
  \semiGermanChords \scm
}
\new ChordNames {
  \set instrumentName = #"italian"
  \italianChords \scm
}
\new ChordNames {
  \set instrumentName = #"french"
  \frenchChords \scm
}
\context Voice { \scm }
>>
```

default	C/C	C#/C#	B/B	B#/B#	Bb/Bb
german	C/c	C#/cis	H/h	H#/his	B/b
semi-german	C/c	C#/cis	H/h	H#/his	Bb/b
italian	Do/Do	Do #/Do #	Si/Si	Si #/Si #	Si b/Si b
french	Do/Do	Do #/Do #	Si/Si	Si #/Si #	Si b/Si b



## Displaying complex chords

Here is a way to display a chord where the same note is played twice with different accidentals.

```
fixA = {
  \once \override Stem.length = #12
}

fixB = {
  \once \override NoteHead.X-offset = #1.7
  \once \override Stem.length = #7
  \once \override Stem.rotation = #'(45 0 0)
  \once \override Stem.extra-offset = #'(-0.1 . -0.2)
  \once \override Flag.style = #'no-flag
  \once \override Accidental.extra-offset = #'(4 . -.1)
}

\relative c' {
  << { \fixA <b d!>8 } \ { \voiceThree \fixB dis } >> s
}
```



## Manually break figured bass extenders for only some numbers

Figured bass often uses extenders to indicate continuation of the corresponding step. LilyPond tries to make extenders as long as possible, which is not always wanted. To break individual extenders, append the modifier \! to a number.

```
bassfigures = \figuremode {
  \set useBassFigureExtenders = ##t
  <6 4>4 <6 4\!> <6 4\!> <6 4\!> |
  <6\! 4\!> <6 4> <6 4\!> <6 4>
}

<<
  \new Staff \relative c'' { c1 c1 }
  \new FiguredBass \bassfigures
>>
```



## Print chord names with same root and different bass as slash and bass note

To print subsequent ChordNames only differing in its bass note as slash and bass note, use the Scheme engraver defined in this snippet. The behaviour may be controlled in detail by the chordChanges context property.

```
#(define Bass_changes_equal_root_engraver
  (lambda (ctx)
    "For sequential `ChordNames` with the same root but a different bass,
    the root markup is dropped: D D/C D/B -> D /C /B.
    The behaviour may be controlled by setting the `chordChanges` context
    property."
    (let ((chord-pitches '())
          (last-chord-pitches '())
          (bass-pitch #f))
      (make-engraver
        ((initialize this-engraver)
         (let ((chord-note-namer (ly:context-property ctx
                                                    'chordNoteNamer)))
           ;; Set 'chordNoteNamer, respect user setting if already done
           (ly:context-set-property! ctx 'chordNoteNamer
                                     (if (procedure? chord-note-namer)
                                         chord-note-namer
                                         note-name->markup))))
          (listeners
            ((note-event this-engraver event)
             (let* ((pitch (ly:event-property event 'pitch))
                    (pitch-name (ly:pitch-notename pitch))
                    (pitch-alt (ly:pitch-alteration pitch))
                    (bass (ly:event-property event 'bass #f))
                    (inversion (ly:event-property event 'inversion #f)))
               ;; Collect notes of the chord
               ;; - to compare inversed chords we need to collect the
               ;;   bass note as usual member of the chord, whereas an
               ;;   added bass must be treated separate from the usual
               ;;   chord-notes
               ;; - notes are stored as pairs containing their
               ;;   pitch-name (an integer), i.e. disregarding their
               ;;   octave and their alteration
               (cond (bass (set! bass-pitch pitch)
                             (inversion
                              (set! bass-pitch pitch)
                              (set! chord-pitches
                                   (cons (cons pitch-name pitch-alt)
                                           chord-pitches))))
                     (else
```

```

        (set! chord-pitches
          (cons (cons pitch-name pitch-alt)
                chord-pitches))))))

(acknowledgers
  ((chord-name-interface this-engraver grob source-engraver)
   (let ((chord-changes (ly:context-property ctx
                                             'chordChanges #f)))
     ;; If subsequent chords are equal apart from their bass,
     ;; reset the 'text-property.
     ;; Equality is done by comparing the sorted lists of this
     ;; chord's elements and the previous chord. Sorting is
     ;; needed because inverted chords may have a different
     ;; order of pitches. `chord-changes` needs to be true.
     (if (and bass-pitch
               chord-changes
               (equal?
                (sort chord-pitches car<)
                (sort last-chord-pitches car<)))
         (ly:grob-set-property!
          grob 'text
          (make-line-markup
           (list
            (ly:context-property ctx 'slashChordSeparator)
            ((ly:context-property ctx 'chordNoteNamer)
             bass-pitch
             (ly:context-property ctx
                                   'chordNameLowercaseMinor))))))
         (set! last-chord-pitches chord-pitches)
         (set! chord-pitches '())
         (set! bass-pitch #f))))

((finalize this-engraver)
 (set! last-chord-pitches '()))))

myChords = \chordmode {
  % \germanChords

  \set chordChanges = ##t
  d2:m d:m/cis

  d:m/c
  \set chordChanges = ##f
  d:m/b

  e1:7
  \set chordChanges = ##t
  e
  \break

  \once \set chordChanges = ##f

```

```

e1/f
e2/gis e/+gis e e:m/f d:m d:m/cis d:m/c
\set chordChanges = ##f
d:m/b
}

<<
\new ChordNames
  \with { \consists #Bass_changes_equal_root_engraver }
  \myChords
\new Staff \myChords
>>

```

Two staves of musical notation. The first staff shows chords: Dm, /C#, /C, Dm/B, E<sup>7</sup>, E. The second staff shows chords: E/F, /G#, E, Em/F, Dm, /C#, /C, Dm/B. The notes are represented by vertical stems with flags, and the chord symbols are placed above them.

## Showing chords at changes

By default, every chord entered is printed. This behavior can be modified so that chord names are printed only at the start of lines or when the chord changes.

```

harmonies = \chordmode {
  c'1:m c:m \break
  c'1:m c:m d
}

<<
\new ChordNames {
  \set chordChanges = ##t
  \harmonies
}
\new Staff {
  \harmonies
}
>>

```

Two staves of musical notation. The first staff shows chords: Cm, Cm. The second staff shows chords: Cm, D. The notes are represented by vertical stems with flags, and the chord symbols are placed above them.

## Simple lead sheet

When put together, chord names, a melody, and lyrics form a lead sheet.

```
<<
\chords { c2 g:sus4 f e }
\new Staff \relative c' {
  a4 e c8 e r4
  b2 c4( d)
}
\addlyrics { One day this shall be free __ }
>>
```



## Single-staff template with notes, lyrics, and chords

This template allows the preparation of a song with melody, words, and chords.

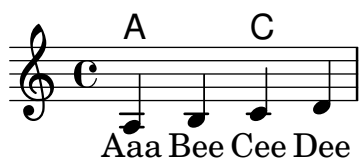
```
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

harmonies = \chordmode {
  a2 c
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \harmonies
    }
    \new Voice = "one" { \autoBeamOff \melody }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
  \midi { }
}
```



## Single-staff template with notes, lyrics, chords, and frets

Here is a simple lead sheet template with melody, lyrics, chords, and fret diagrams.

```

verseI = \lyricmode {
  \set stanza = #"1."
  This is the first verse
}

verseII = \lyricmode {
  \set stanza = #"2."
  This is the second verse.
}

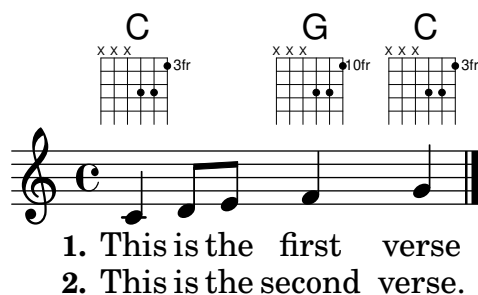
theChords = \chordmode {
  % insert chords for chordnames and fretboards here
  c2 g4 c
}

staffMelody = \relative c' {
  \key c \major
  \clef treble
  % Type notes for melody here
  c4 d8 e f4 g
  \bar "|"
}

\score {
  <<
    \context ChordNames { \theChords }
    \context FretBoards { \theChords }
    \new Staff {
      \context Voice = "voiceMelody" { \staffMelody }
    }
    \new Lyrics = "lyricsI" {
      \lyricsto "voiceMelody" \verseI
    }
    \new Lyrics = "lyricsII" {
      \lyricsto "voiceMelody" \verseII
    }
  >>
  \layout { }
  \midi { }
}

```





1. This is the first verse  
2. This is the second verse.

## Single-staff template with notes and chords

Want to prepare a lead sheet with a melody and chords? Look no further!

```
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  f4 e8[ c] d4 g |
  a2 ~ a
}

harmonies = \chordmode {
  c4:m f:min7 g:maj c:aug |
  d2:dim b4:5 e:sus
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \harmonies
    }
    \new Staff \melody
  >>
  \layout{ }
  \midi { }
}
```



## Vertically centering paired figured bass extenders

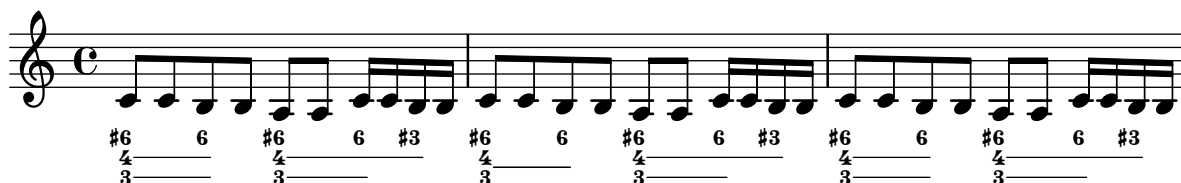
Where figured bass extender lines are being used by setting `useBassFigureExtenders` to `#t`, pairs of congruent figured bass extender lines are vertically centered if `figuredBassCenterContinuations` is set to `#t`.

```
<<
  \relative c' {
    \repeat unfold 3 {
      c8 c b b a a c16 c b b
    }
  }
```

```

}
\figures {
  \set useBassFigureExtenders = ##t
  <6+ 4 3>4 <6 4 3>8 r
  <6+ 4 3>4 <6 4 3>8 <4 3+>16 r
  \set figuredBassCenterContinuations = ##t
  <6+ 4 3>4 <6 4 3>8 r
  <6+ 4 3>4 <6 4 3>8 <4 3+>16 r
  \set figuredBassCenterContinuations = ##f
  <6+ 4 3>4 <6 4 3>8 r
  <6+ 4 3>4 <6 4 3>8 <4 3+>16 r
}
>>

```



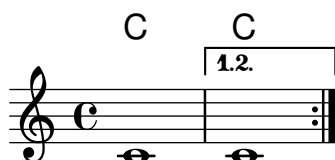
## Volta below chords

By adding the `Volta_engraver` to the relevant staff, volte can be put below chords.

```

\score {
  <<
    \chords { c1 c1 }
    \new Staff \with { \consists "Volta_engraver" }
    {
      \repeat volta 2 { c'1 \alternative { c' } }
    }
  >>
  \layout {
    \context {
      \Score
      \remove "Volta_engraver"
    }
  }
}

```



## 16 Contemporary music

See also Section “Contemporary music” in *Notation Reference*.

### Beam nibs

Beam nibs at the start and end of beams together with beams attached to solitary notes that look like flat flags are possible with a combination of `stemLeftBeamCount`, `stemRightBeamCount`, and paired `[]` beam indicators.

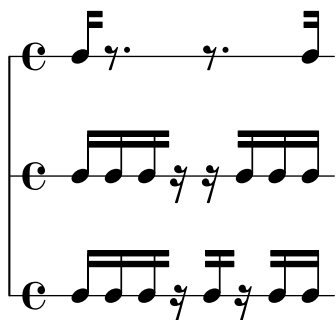
For imitating right-pointing flat flags on lone notes, use paired `[]` beam indicators and set `stemLeftBeamCount` to zero. For imitating left-pointing flat flags on lone notes, set `stemRightBeamCount` to zero instead (line one).

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

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

Note that `\set stemLeftBeamCount` is always equivalent to `\once \set`. In other words, the beam count settings are not “sticky”, so the pair of nibs attached to the lone 16th note in the last example has nothing to do with the `\set` command for the beam before.

```
\score {
  <<
    \new RhythmicStaff {
      \set stemLeftBeamCount = 0
      c16[] r8.
      r8.
      \set stemRightBeamCount = 0
      16[]
    }
    \new RhythmicStaff {
      16 16
      \set stemRightBeamCount = 2
      16 r r
      \set stemLeftBeamCount = 2
      16 16 16
    }
    \new RhythmicStaff {
      16 16
      \set stemRightBeamCount = 2
      16 r16
      16[] r16
      \set stemLeftBeamCount = 2
      16 16
    }
  >>
}
```



## Broken crescendo hairpin

In order to make parts of a crescendo hairpin invisible, the following method is used: A white rectangle is drawn on top of the respective part of the crescendo hairpin, making it invisible. The rectangle is defined as a text markup.

The markup command `with-dimensions` tells LilyPond to consider only the bottom edge of the rectangle when spacing it against the hairpin. The property `staff-padding` prevents the rectangle from fitting between the hairpin and staff.

Make sure the hairpin is in a lower layer than the text markup to draw the rectangle over the hairpin.

```
\relative c' {
  <<
  {
    \dynamicUp
    r2 r16 c'8.\pp r4
  }
  \\\
  {
    \override DynamicLineSpanner.layer = #0
    des,2\mf\< ~
    \override TextScript.layer = #2
    \once\override TextScript.staff-padding = #6
    \once\override TextScript.vertical-skylines = #'()
    des16_\markup \with-dimensions #'(2 . 7) #'(0 . 0)
      \with-color #white
      \filled-box #'(2 . 7) #'(0 . 2) #0
    r8. des4 ~ des16->\sff r8.
  }
  >>
}
```



## Changing time signatures inside a polymetric section using \scaleDurations

***Flexible polymeric with unaligned measures***

To support explicit creation of independently measured contexts, remove the `Timing_translator` from `Score` context and define a `TimingStaffGroup` context that has `Timing_translator`. This makes `Timing` an alias for `TimingStaffGroup`, targeting `\time` commands to the enclosing `TimingStaffGroup`.

Unlike LilyPond's built-in `\enablePerStaffTiming` command, this approach requires the explicit creation of `TimingStaffGroup` contexts; in exchange, it allows creating multiple `Staff` contexts that jointly follow the measure defined in their enclosing `TimingStaffGroup`.

***Locally scaled time signatures***

Use the unscalable `\time` command to establish a measure of the desired length in `Timing`, a.k.a. `TimingStaffGroup`. In this snippet, all staves below `TimingStaffGroup` use a scaled time signature, so any time signature with the desired measure length is as good as any other. If there were an enclosed context that did not use a scaled time signature, the choice of time signature to set in `Timing` would matter in that context.

Use the `\polymetric \time` command to set scalable metric properties in contexts below `Timing`, and use the `\scaleDurations` command to scale both the local meter and the notes to fit the measure.

```
\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \accepts TimingStaffGroup
  }
  \context {
    \StaffGroup
    \name TimingStaffGroup
    \alias StaffGroup
    \consists "Timing_translator"
  }
}

<<
\new TimingStaffGroup <<
  \new Staff {
    \scaleDurations 8/5 {
      \time 6/5 % to set measure length in Timing
      \context Staff \polymetric \time 6/8
      b8 b b b b b
      \time 4/5 % to set measure length in Timing
      \context Staff \polymetric \time 2/4
      b4 b
    }
  }
}
>>
\new TimingStaffGroup <<
  \new Staff {
    \clef bass
    \time 2/4
    c2 d e f
  }
}
```

```
>>
>>
```



## Clusters

Clusters are a device to denote that a complete range of notes is to be played.

```
fragment = \relative c' {
  c4 f <e d'>4
  <g a>8 <e a> a4 c2 <d b>4
  e2 c
}
```

```
<<
  \new Staff \fragment
  \new Staff \makeClusters \fragment
>>
```



## 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 "|"
}
```



## Flat ties

This snippet provides a function `flared-tie` to draw a tie that consist of straight lines. It is intended as a replacement for the default tie-drawing function (i.e., a replacement argument for the `stencil` property of the `Tie` grob).

The argument of `flared-tie` is a list of coordinate pairs that specify additional points between the first and last point to span up the tie's lines. The first and last point are identical to the original tie's start and end point, respectively. The X and Y coordinate values are multiples of the bounding box length and height of the original tie (also taking care of the tie's direction); consequently, the first point has coordinates (0,0), and the last point (1,0).

The function `flare-tie` defines a shorthand for a flat tie. Further tweaking of the shape is possible by overriding `Tie.details.height-limit` or with `\shape`. It is also possible to change the custom definition on the fly.

```
#(define ((flared-tie coords) grob)
  (define (pair-to-list pair)
    (list (car pair) (cdr pair)))

  (define (normalize-coords goods x y dir)
    (map
     (lambda (coord)
       (cons (* x (car coord)) (* y dir (cdr coord)))))
     goods))

  (define (my-c-p-s points thick)
    (make-connected-path-stencil points thick 1.0 1.0 #f #f))

  ;; Calling `ly:tie::print` and assigning its return value to a
  ;; variable in this outer `let` triggers LilyPond to position the
  ;; tie, allowing us to extract its extents. We only proceed,
  ;; however, if the tie doesn't get discarded (for whatever reason).
  (let ((sten (ly:tie::print grob)))
    (if (grob::is-live? grob)
        (let* ((layout (ly:grob-layout grob))
               (line-thickness (ly:output-def-lookup layout
                                                         'line-thickness))
               (thickness (ly:grob-property grob 'thickness 0.1))
               (used-thick (* line-thickness thickness))
               (dir (ly:grob-property grob 'direction))
               (xex (ly:stencil-extent sten X))
               (yex (ly:stencil-extent sten Y))
               (lenx (interval-length xex))
               (leny (interval-length yex))
               (xtrans (car xex))
               (ytrans (if (> dir 0) (car yex) (cdr yex))))
          ;; Add last point.
          (coord-list (append coords '((1.0 . 0.0))))
          (uplist
           (map pair-to-list
                (normalize-coords coord-list lenx (* leny 2) dir))))
        (ly:stencil-translate
         (my-c-p-s uplist used-thick)
         (cons xtrans ytrans))))
```

```

'()))))

% Define a default tie shape consisting of three straight lines.
#(define flare-tie
  (flared-tie '((0.1 . 0.3) (0.9 . 0.3))))

\relative c' {
  a4~ a
  \once \override Tie.stencil = #flare-tie
  a4~ a \break

  <a c e a c e a c e>~ q
  \once \override Tie.stencil = #flare-tie
  q~ q\break

  <>~\markup \small \typewriter "height-limit = 14"
  \override Tie.details.height-limit = 14
  a'4~ a
  \once \override Tie.stencil = #flare-tie
  a4~ a \break

  <>~\markup \small \typewriter "height-limit = 0.5"
  \override Tie.details.height-limit = 0.5
  a4~ a
  \once \override Tie.stencil = #flare-tie
  a4~ a \break

  \revert Tie.details.height-limit

  <>~\markup \small \typewriter
    "\shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0))"
  \shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0)) Tie
  a4~ a
  \once \override Tie.stencil = #flare-tie
  \shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0)) Tie
  a4~ a \break

  <>~\markup \small \typewriter
    "#(flared-tie '((0.2 . 2) (0.5 . -3) (0.8 . 1)))"
  \once \override Tie.stencil =
    #(flared-tie '((0.2 . 2) (0.5 . -3) (0.8 . 1)))
  a4~ a
  <>~\markup \small \typewriter
    "#(flared-tie '((0.5 . 2)))"
  \once \override Tie.stencil = #(flared-tie '((0.5 . 2)))
  a'4~ a
}

```





2

3 height-limit = 14

4 height-limit = 0.5

5 \shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0))  
#(flared-tie '((0.2 . 2) (0.5 . -3) (0.8 . 1)))

6 #(flared-tie '((0.5 . 2)))

## Flute slap notation

It is possible to indicate special articulation techniques such as a flute “tongue slap” by replacing the note head with the appropriate glyph. For that we can draw the accent-like note head with `\markup`.

```
slap =
#(define-music-function (music) (ly:music?)
  #{
    \temporary \override NoteHead.stencil =
      #ly:text-interface::print
    \temporary \override NoteHead.text =
      \markup
        \translate #'(1 . 0)
        \override #'(thickness . 1.4)
        \overlay { \draw-line #'(-1.2 . 0.4)
                  \draw-line #'(-1.2 . -0.4) }
    \temporary \override NoteHead.stem-attachment =
      #(lambda (grob)
        (let* ((stem (ly:grob-object grob 'stem))
              (dir (ly:grob-property stem 'direction UP))
              (is-up (eqv? dir UP)))
          (cons dir (if is-up 0 -0.8)))))
    #music
    \revert NoteHead.stencil
    \revert NoteHead.text
    \revert NoteHead.stem-attachment
  })
```

```
\relative c' {
  c4 \slap c d r
  \slap { g4 a } b r
}
```



## Heavily customized polymetric time signatures

Though the polymetric time signature shown is not the most essential item here, it has been included to show the beat of this piece (which is the template of a real Balkan song, by the way).

```
melody = \relative c'' {
  \key g \major
  \compoundMeter #'((3 8) (2 8) (2 8) (3 8) (2 8) (2 8)
                    (2 8) (2 8) (3 8) (2 8) (2 8))
  \set Timing.beamExceptions = #'()
  \set Timing.beatStructure = 3,2,2,3,2,2,2,2,3,2,2
  c8 c c d4 c8 c b c b a4 g fis8 e d c b' c d e4-^ fis8 g \break
  c,4. d4 c4 d4. c4 d c2 d4. e4-^ d4
  c4. d4 c4 d4. c4 d c2 d4. e4-^ d4 \break
}

drum = \new DrumStaff \drummode {
  \repeat volta 2 {
    bd4.^ \markup { Drums } sn4 bd \bar ";"
    sn4. bd4 sn \bar ";"
    bd sn bd4. sn4 bd
  }
}

\new Staff {
  \melody
  \drum
}
```

## Marking notes of spoken parts with a cross on the stem (Sprechstimme)

This example shows how to put crosses on stems. Mark the beginning of a spoken section with the command `\speakOn` and end it with `\speakOff`.

```
speakOn = \override Stem.stencil =
  #(lambda (grob)
    (let* ((x-parent (ly:grob-parent grob X))
      (is-rest? (ly:grob? (ly:grob-object x-parent 'rest))))
      (if is-rest?
        empty-stencil
        (ly:stencil-combine-at-edge
          (ly:stem::print grob)
          Y
          (- (ly:grob-property grob 'direction))
          (grob-interpret-markup
            grob
            (markup #:center-align #:fontsize -4
              #:musicglyph "noteheads.s2cross")))
          -1.7))))
```

```
speakOff = \revert Stem.stencil
```

```
\new Staff {
  \relative c'' {
    a4 b a c
    \speakOn
    g4 f r g8 a
    b4 r r8 d e4
    \speakOff
    c4 a g f
  }
}
```



## Non-traditional key signatures

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

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

```
\set Staff.keyAlterations =
  #`(((octave . step) . alter) ((octave . step) . alter) ...)
```

where, for each element in the list, *octave* specifies the octave (0 being the octave from middle C to the B above), *step* specifies the note within the octave (0 means C and 6 means B), and *alter* is one of SHARP, FLAT, DOUBLE-SHARP, etc., preceded by a comma.

Alternatively, you can use the more concise format *(step . alter)* for each item in the list if the same alterations are used 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.

```
\include "arabic.ly"
```

```

\relative do' {
  \set Staff.keyAlterations = #`((0 . ,SEMI-FLAT)
                                (1 . ,SEMI-FLAT)
                                (2 . ,FLAT)
                                (5 . ,FLAT)
                                (6 . ,SEMI-FLAT))

  % \set Staff.extraNatural = ##f
  re reb \down reb resd
  dod dob dosd \down dob |
  dobsb dodsd do do |
}

```



## Printing music with different time signatures

In the following snippet, two parts have a completely different time signature, yet remain synchronized.

The bar lines can no longer be printed at the Score level; to allow independent bar lines in each part, the `Default_barline_engraver` and `Timing_translator` are moved from the `Score` context to the `Staff` context.

If bar numbers are required, the `Bar_number_engraver` should also be moved, since it relies on properties set by the `Timing_translator`; a `\with` block can be used to add bar numbers to the relevant staff.

```

global = {
  \time 3/4 s2.*3 \break
  s2.*3
}

\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Bar_number_engraver"
    \override SpacingSpanner.uniform-stretching = ##t
    \override SpacingSpanner.strict-note-spacing = ##t
    \proportionalNotationDuration = #1/64
  }
  \context {
    \Staff
    \consists "Timing_translator"
  }
  \context {
    \Voice
    \remove "Forbid_line_break_engraver"
    \tupletFullLength = ##t
  }
}

```

```

Bassklarinette = \new Staff \with {
  \consists "Bar_number_engraver"
  barNumberVisibility = #(every-nth-bar-number-visible 2)
  \override BarNumber.break-visibility = #end-of-line-invisible
} <<
\global
{
  \clef treble
  \time 3/8 d''4. |
  \time 3/4 r8 des''2( c''8) |
  \time 7/8 r4. ees''2 ~ |
  \time 2/4 \tupletUp \tuplet 3/2 { ees''4 r4 d''4 ~ } |
  \time 3/8 \tupletUp \tuplet 4/3 { d''4 r4 } |
  \time 2/4 e''2 |
  \time 3/8 es''4. |
  \time 3/4 r8 d''2 r8 |
}
>>

Perkussion = \new StaffGroup <<
  \new Staff <<
    \global
    {
      \clef percussion
      \time 3/4 r4 c'2 ~ |
      c'2. |
      R2. |
      r2 g'4 ~ |
      g'2. ~ |
      g'2. |
    }
  >>
  \new Staff <<
    \global {
      \clef percussion
      \time 3/4 R2. |
      g'2. ~ |
      g'2. |
      r4 g'2 ~ |
      g'2 r4 |
      g'2. |
    }
  >>
>>

\score {
  <<
    \Bassklarinette
    \Perkussion
  >>
}

```

The image displays a musical score for a piece titled 'Screech and Boink'. The score is written for a piano, featuring a treble staff and two bass staves. The notation is highly complex and contemporary, characterized by frequent changes in time signature and key signature. The score is divided into three systems, each starting with a measure number in parentheses: (1), (4), and 8. The first system (measures 1-3) begins in 3/8 time, changes to 3/4, then 7/8, and finally 2/4. The second system (measures 4-6) continues with 3/8, 3/4, and 3/8 time signatures. The third system (measures 8-9) is in 3/4 time. The notation includes various rhythmic values, including dotted notes, rests, and beams, as well as dynamic markings like 'p' (piano). The overall style is experimental and non-traditional.

## Screech and Boink

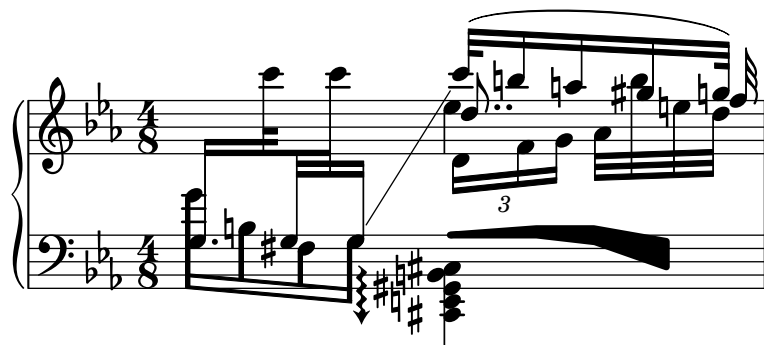
Random complex notation.

```
\score {
  \context PianoStaff <<
    \new Staff = "up" {
      \time 4/8
      \key c \minor
      <<
        {
          \revert Stem.direction
          \change Staff = down
          \set subdivideBeams = ##t
          g16.[
          \change Staff = up
          c'''32
          \change Staff = down
          g32
          \change Staff = up
          c'''32
        }
      }
    }
}
```

```

        \change Staff = down
        g16]
        \change Staff = up
        \stemUp
        \set followVoice = ##t
        c''32([ b''16 a''16 gis''16 g''32)]
    }
    \\\
    { s4 \tuplet 3/2 { d'16[ f' g'] } as'32[ b''32 e'' d''] }
    \\\
    { s4 \autoBeamOff d''8.. f''32 }
    \\\
    { s4 es''4 }
    >>
}
\new Staff = "down" {
    \clef bass
    \key c \minor
    \set subdivideBeams = ##f
    \override Stem.french-beaming = ##t
    \override Beam.beam-thickness = #0.3
    \override Stem.thickness = #4.0
    g'16[ b16 fis16 g16]
    <<
        \makeClusters {
            as16 <as b> <g b> <g cis>
        }
        \\\
        {
            \override Staff.Arpeggio.arpeggio-direction = #DOWN
            <cis, e, gis, b, cis>4\arpeggio
        }
    >>
}
>>
\midi {
    \tempo 8 = 60
}
\layout {
    ragged-right = ##t
    \context {
        \Staff
        \consists "Horizontal_bracket_engraver"
    }
}
}

```



## Stemlets

In some notational conventions beams are allowed to extend over rests. Depending on preference, these beams may drop ‘stemlets’ to help the eye appreciate the rhythm better, and in some modern music the rest itself is omitted and only the stemlet remains.

This snippet shows a progression from traditional notation, to beams over the rest, to stemlets over the rest, to stemlets alone. Stemlets are generated by overriding the `stemlet-length` property of `Stem`, and rests are hidden by using `\hide`.

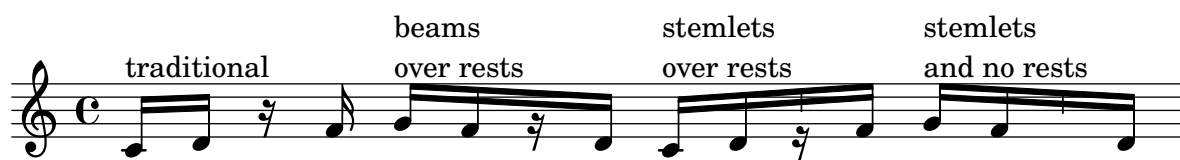
Some `\markup` elements are included in the source to highlight the different notations.

```
\paper {
  ragged-right = ##f
}

{
  c'16~\markup { traditional } d' r f'
  g'16[~\markup \column { "beams" "over rests" } f' r d']

  % N.B. use Score.Stem to set for the whole score.
  \override Staff.Stem.stemlet-length = #0.75

  c'16[~\markup \column { "stemlets" "over rests" } d' r f']
  g'16[~\markup \column { "stemlets" "and no rests" } f'
  \once \hide Rest
  r16 d']
}
```





## 17 Ancient notation

See also Section “Ancient notation” in *Notation Reference*.

### Adding a figured bass above or below the notes

When writing figured bass, you can place the figures above or below the bass notes by using the commands `\bassFigureStaffAlignmentDown` and `\bassFigureStaffAlignmentUp`. Prepend `\once` to the command if you want to modify only the next figured bass.

The command `\bassFigureStaffAlignmentNeutral` resets the direction of figured bass to the default value.

```
bass = {
  \clef bass
  g4 b, c d |
  e d8 c d2
}

continuo = \figuremode {
  <_>4 <6>4 <5/>4
  \bassFigureStaffAlignmentUp
  <_+>4 <6> |
  \set Staff.useBassFigureExtenders = ##t
  \bassFigureStaffAlignmentDown
  <4>4. <4>8 <_+>4
}

\score {
  <<
    \new Staff = bassStaff \bass
    \context Staff = bassStaff \continuo
  >>
}
```



### Ancient fonts

This snippets shows many of the symbols contained in the Emmentaler font that are used by LilyPond for typesetting ancient notation.

```
m = { c1 e f ges cis' \bar "||" }
```

```
\markup \with-true-dimensions % work around a cropping issue
\score {
  \new VaticanaVoice {
    \clef "vaticana-fa2"
    \key es \major
    \textMark \markup \rounded-box "Vaticana clefs, custos and note heads"

    \override NoteHead.style = #'vaticana.punctum
```

```

<>^"vaticana.punctum" \m

\override NoteHead.style = #'vaticana.inclinatum
<>^"vaticana.inclinatum" \m

\override NoteHead.style = #'vaticana.quilisma
<>^"vaticana.quilisma" \m

\clef "vaticana-fa1"
\override NoteHead.style = #'vaticana.plica
<>^"vaticana.plica" \m

\override NoteHead.style = #'vaticana.reverse.plica
<>^"vaticana.reverse.plica" \m

\override NoteHead.style = #'vaticana.punctum.cavum
<>^"vaticana.punctum.cavum" \m

\override NoteHead.style = #'vaticana.lpes
<>^"vaticana.punctum.lpes" \m

\override NoteHead.style = #'vaticana.upes
<>^"vaticana.punctum.upes" \m

\override NoteHead.style = #'vaticana.vupes
<>^"vaticana.punctum.vupes" \m

\override NoteHead.style = #'vaticana.linea.punctum
<>^"vaticana.punctum.linea" \m

\override NoteHead.style = #'vaticana.epiphonus
<>^"vaticana.punctum.epiphonus" \m

\override NoteHead.style = #'vaticana.cephalicus
<>^"vaticana.punctum.cephalicus" \m

\break

\textMark \markup \rounded-box "Medicaea clefs, custos and note heads"
\set VaticanaStaff.alterationGlyphs =
  #alteration-medicaea-glyph-name-alist
\override VaticanaStaff.Custos.style = #'medicaea

\clef "medicaea-fa2"
\override NoteHead.style = #'medicaea.punctum
<>^"medicaea.punctum" \m

\clef "medicaea-do2"
\override NoteHead.style = #'medicaea.inclinatum
<>^"medicaea.inclinatum" \m

\override NoteHead.style = #'medicaea.virga

```

```

<>^"medicaea.virga" \m

\clef "medicaea-fa1"
\override NoteHead.style = #'medicaea.rvirga
<>^"medicaea.rvirga" \m

\break

\textMark \markup \rounded-box "Hufnagel clefs, custos and note heads"
\set Staff.alterationGlyphs =
  #alteration-hufnagel-glyph-name-alist
\override VaticanaStaff.Custos.style = #'hufnagel
\clef "hufnagel-fa2"

\break

\override NoteHead.style = #'hufnagel.punctum
<>^"hufnagel.punctum" \m

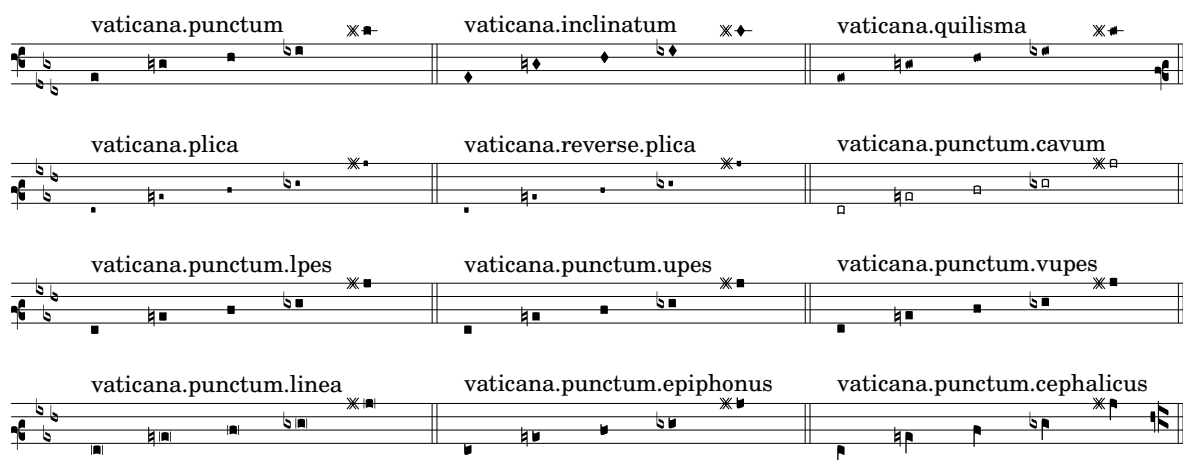
\clef "hufnagel-do2"
\override NoteHead.style = #'hufnagel.lpes
<>^"hufnagel.lpes" \m

\clef "hufnagel-do-fa"
\override NoteHead.style = #'hufnagel.virga
<>^"hufnagel.virga" \m
}

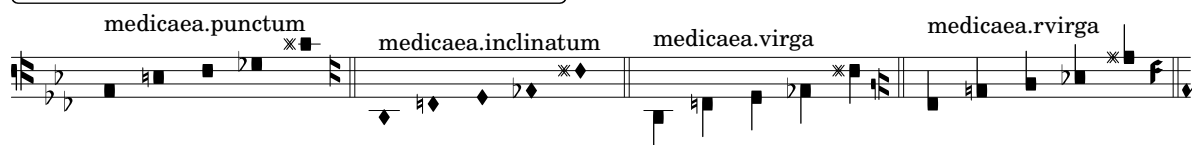
\layout {
  \context {
    \Score
    \override TextScript.font-size = #-2
    \override TextMark.break-align-symbols = #'(left-edge clef staff-bar)
    \override TextMark.padding = 4
    \omit BarNumber
  }
  \context {
    \VaticanaStaff
    alterationGlyphs =
      #alteration-vaticana-glyph-name-alist
  }
}
}

```

### Vaticana clefs, custos and note heads



### Medicaea clefs, custos and note heads



### Hufnagel clefs, custos and note heads



## 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.

```
chant = \relative c' {
  \set Score.timing = ##f
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g f) a2 \finalis \break
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g a) g2( f) \finalis
}

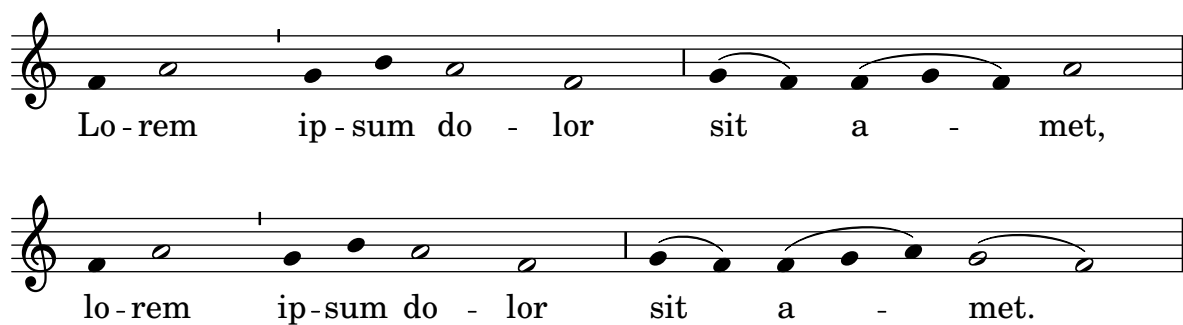
verba = \lyricmode {
  Lo -- rem ip -- sum do -- lor sit a -- met,
  lo -- rem ip -- sum do -- lor sit a -- met.
}
```

```
\score {
  \new GregorianTranscriptionStaff <<
```

```

\new GregorianTranscriptionVoice = "melody" \chant
\new GregorianTranscriptionLyrics = "one" \lyricsto melody \verba
>>
}

```



## Ancient time signatures

Time signatures may also be engraved in an old style.

```

{
  \override Staff.TimeSignature.style = #'neomensural
  s1
}

```



## Chant or psalm notation

This form of notation is used for psalm chant, where verses aren't always of the same length.

```

stemOff = \hide Staff.Stem
stemOn  = \undo \stemOff

\score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \key g \minor
    \cadenzaOn
    \stemOff a'\breve bes'4 g'4
    \stemOn a'2 \section
    \stemOff a'\breve g'4 a'4
    \stemOn f'2 \section
    \stemOff a'\breve~\markup { \italic flexe }
    \stemOn g'2 \fine
  }
}

```



## Custodes

Custodes may be engraved in various styles.

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

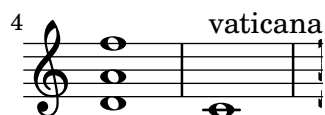
\markup \with-true-dimensions % work around a cropping issue
\score {
  \new Staff \with { \consists "Custos_engraver" } \relative c' {
    \override Staff.Custos.neutral-position = #4

    \override Staff.Custos.style = #'hufnagel
    c1^"hufnagel" \break
    <d a' f'>1

    \override Staff.Custos.style = #'medicaea
    c1^"medicaea" \break
    <d a' f'>1

    \override Staff.Custos.style = #'vaticana
    c1^"vaticana" \break
    <d a' f'>1

    \override Staff.Custos.style = #'mensural
    c1^"mensural" \break
    <d a' f'>1
  }
}
```



## Incipit

When transcribing mensural music, an incipit at the beginning of the piece is useful to indicate the original key and tempo. While today musicians are used to bar lines in order to faster recognize rhythmic patterns, bar lines were not yet invented during the period of mensural music; in fact, the meter often changed after every few notes. As a compromise, bar lines are often printed between the staves rather than on the staves.

*% A short excerpt from the Jubilate Deo by Orlande de Lassus*

```
global = {
  \set Score.skipBars = ##t
  \key g \major
  \time 4/4

  % the actual music
  \skip 1*8

  % let finis bar go through all staves
  \override Staff.BarLine.transparent = ##f

  % finis bar
  \bar "|."
}

discantusIncipit = \new PetrucciStaff {
  \clef "petrucci-c1"
  \key f \major
  \time 2/2
  c''1.
}

discantusNotes = {
  \transpose c' c'' {
    \clef "treble"
    d'2. d'4 |
    b e' d'2 |
    c'4 e'4.( d'8 c' b |
    a4) b a2 |
    b4.( c'8 d'4) c'4 |
    \once \hide NoteHead
    c'1 |
    b\breve |
  }
}

discantusLyrics = \lyricmode {
  Ju -- bi -- la -- te De -- o,
  om -- nis ter -- ra, __ om-
  "...
  -us.
}
```

```
altusIncipit = \new PetrucciStaff {
  \clef "petrucci-c3"
  \key f \major
  \time 2/2
  e'1\rest f'1.
}
```

```
altusNotes = {
  \transpose c' c'' {
    \clef "treble"
    r2 g2. e4 fis g |
    a2 g4 e |
    fis g4.( fis16 e fis4) |
    g1 |
    \once \hide NoteHead
    g1 |
    g\breve |
  }
}
```

```
altusLyrics = \lyricmode {
  Ju -- bi -- la -- te
  De -- o, om -- nis ter -- ra,
  "...
  -us.
}
```

```
tenorIncipit = \new PetrucciStaff {
  \clef "petrucci-c4"
  \key f \major
  \time 2/2
  r\longa
  r\breve
  r1 c'1.
}
```

```
tenorNotes = {
  \transpose c' c' {
    \clef "treble_8"
    R1 |
    R1 |
    R1 |
    % two measures
    r2 d'2. d'4 b e' |
    \once \hide NoteHead
    e'1 |
    d'\breve |
  }
}
```

```
tenorLyrics = \lyricmode {
  Ju -- bi -- la -- te
```



```

    "...
    -us.
}

bassusIncipit = \new PetrucciStaff {
  % The original print shows the b flat
  % for the f major key signature twice.
  \override Staff.KeySignature.flat-positions = #'((-7 . 6))
  \clef "mensural-f"
  \key f\major
  \time 2/2
  \tweak Y-offset #1 r\longa \tweak Y-offset #1 r\longa
  f1.
}

bassusNotes = {
  \transpose c' c' {
    \clef "bass"
    R1 |
    R1 |
    R1 |
    R1 |
    g2. e4 |
    \once \hide NoteHead
    e1 |
    g\breve |
  }
}

bassusLyrics = \lyricmode {
  Ju -- bi-
  "...
  -us.
}

\score {
  <<
  \new StaffGroup = choirStaff <<
  \new Voice = "discantusNotes" <<
    \set Staff.instrumentName = "Discantus"
    \incipit #1 \discantusIncipit
    \global
    \discantusNotes
  >>
  \new Lyrics \lyricsto discantusNotes { \discantusLyrics }
  \new Voice = "altusNotes" <<
    \set Staff.instrumentName = "Altus"
    \global
    \incipit #1 \altusIncipit
    \altusNotes
  >>
  \new Lyrics \lyricsto altusNotes { \altusLyrics }
}

```

```

\new Voice = "tenorNotes" <<
  \set Staff.instrumentName = "Tenor"
  \global
  \incipit #1 \tenorIncipit
  \tenorNotes
>>
\new Lyrics \lyricsto tenorNotes { \tenorLyrics }
\new Voice = "bassusNotes" <<
  \set Staff.instrumentName = "Bassus"
  \global
  \incipit #1 \bassusIncipit
  \bassusNotes
>>
\new Lyrics \lyricsto bassusNotes { \bassusLyrics }
>>
>>
\layout {
  \context {
    \Score
    %% no bar lines in staves or lyrics
    \hide BarLine
  }
  %% the next two instructions keep the lyrics between the bar lines
  \context {
    \Lyrics
    \consists "Bar_engraver"
    \consists "Separating_line_group_engraver"
  }
  \context {
    \Voice
    %% no slurs
    \hide Slur
    %% Comment in the below "\remove" command to allow line
    %% breaking also at those bar lines where a note overlaps
    %% into the next measure. The command is commented out in this
    %% short example score, but especially for large scores, you
    %% will typically yield better line breaking and thus improve
    %% overall spacing if you comment in the following command.
    %%\remove "Forbid_line_break_engraver"
  }
  indent = 5\cm
  incipit-width = 2.5\cm
}
}

```

Discantus

Altus

Tenor

Bassus

Ju - bi - la - te De - o, om -

Ju - bi - la - te De - o, om -

Ju - bi - la - te De - o, om -

Ju - bi - la - te De - o, om -

- nis ter - ra, om- ... -us.

nis ter - ra, ... -us.

nis ter - ra, ... -us.

Ju - bi - la - te ... -us.

Ju - bi - la - te ... -us.

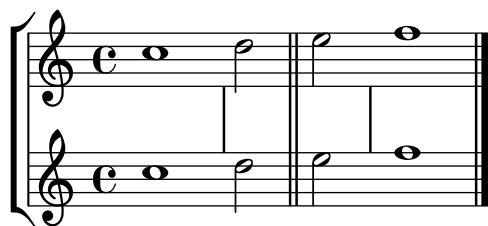
## Mensurstriche layout (bar lines between the staves)

*Mensurstriche*, bar lines between but not through staves, can be printed by setting `measureBarType` to `"-span|"` and using a grouping context that allows span bars, such as `StaffGroup`.

```
\layout {
  \context {
    \Staff
    measureBarType = "-span|"
  }
}
```

```
music = \fixed c'' {
  c1
  d2 \section e2
  f1 \fine
}
```

```
\new StaffGroup <<
  \new Staff \music
  \new Staff \music
>>
```



## Rest styles

Rests may be used in various styles.

```
restsA = {
  r\maxima r\longa r\breve r1 r2 r4 r8 r16 s32
  s64 s128 s256 s512 s1024 s1024
}
restsB = {
  r\maxima r\longa r\breve r1 r2 r4 r8 r16 r32
  r64 r128 r256 r512 r1024 s1024
}

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

  \override Staff.Rest.style = #'mensural
  <>\markup \typewriter { mensural } \restsA \bar "" \break

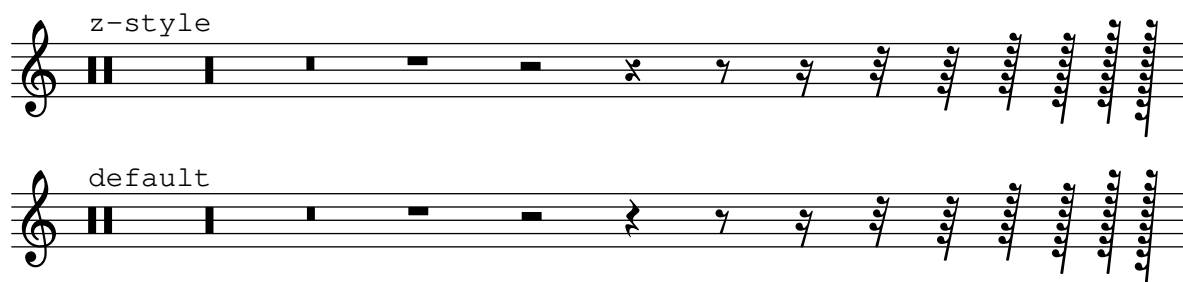
  \override Staff.Rest.style = #'neomensural
  <>\markup \typewriter { neomensural } \restsA \bar "" \break

  \override Staff.Rest.style = #'classical
  <>\markup \typewriter { classical } \restsB \bar "" \break

  \override Staff.Rest.style = #'z
  <>\markup \typewriter { z-style } \restsB \bar "" \break

  \override Staff.Rest.style = #'default
  <>\markup \typewriter { default } \restsB \bar "" \break
}
```





## Using tags to produce mensural and modern music from the same source

Using tags it is possible to produce both mensural and modern notation from the same music. In this snippet, a function `\menrest` is introduced, allowing mensural rests to be pitched as in the original, but with modern rests in the standard staff position.

Tags can also be used where other differences are needed: for example using “whole measure rests” (`R1`, `R\breve`, etc.) in modern music, but normal rests (`r1`, `r\breve`, etc.) in the mensural version. Converting mensural music to its modern equivalent is usually referred to as *transcription*.

The call `c4.\Be c8 c\Am` is the same as `c4.[ c8 c]`. However, it suppresses warnings if it starts on a note that can’t hold a beam but needs it anyway due to the use of `Completion_heads_engraver`.

[Note that the custos sticks out into the right margin and might be cut off if the LilyPond output gets cropped tightly. The use of `\with-true-dimensions` below avoids this.]

```
\layout {
  line-width = 150\mm
}

menrest = #(define-music-function (note) (ly:music?)
  #{
    \tag #'mens $(make-music 'RestEvent note)
    \tag #'mod $(make-music 'RestEvent note 'pitch '())
  #})

Be = \tag #'mod
  #(begin
    (ly:expect-warning (G_ "stem does not fit in beam"))
    (ly:expect-warning (G_ "beam was started here"))
    (make-span-event 'BeamEvent START))

Am = \tag #'mod ]

MenStyle = {
  \override Score.BarNumber.transparent = ##t
  \override Stem.neutral-direction = #up
  \omit Slur
  \omit Beam
}

finalis = \section

Music = \relative c'' {
  \key f \major
```

```

g1 d'2 \menrest bes4 bes a2 \menrest r4 g4 fis4.
fis8 fis4 fis g e f4.([ g8] a4[ g8 f]
g2.\Be fis8 e\Am fis2) g\breve \finalis
}

MenLyr = \lyricmode {
  So farre, deere life, deare life,
  from thy bright beames ab- en- ted,
}
ModLyr = \lyricmode {
  So far, dear life, dear life,
  from your bright beams ab -- sen -- ted, __
}

\markup \with-true-dimensions % work around a cropping issue
\score {
  \keepWithTag #'mens {
    <<
    \new PetrucciStaff {
      \new PetrucciVoice = "Cantus" {
        \clef "petrucci-c1" \time 4/4 \MenStyle \Music
      }
    }
    \new Lyrics \lyricsto "Cantus" \MenLyr
  } >>
}
\layout {
  \context {
    \PetrucciVoice
    % No longer necessary starting with version 2.25.23.
    \override Flag.style = #'mensural
  }
}

\markup\vspace #1

\score {
  \keepWithTag #'mod {
    \new ChoirStaff <<
    \new Staff {
      \new Voice = "Sop" \with {
        \remove "Note_heads_engraver"
        \consists "Completion_heads_engraver"
        \remove "Rest_engraver"
        \consists "Completion_rest_engraver"
      } \shiftDurations 1 0 { \time 2/4 \autoBeamOff \Music }
    }
    \new Lyrics \lyricsto "Sop" \ModLyr
  } >>
}

```

So farre, deere life, deare life, from thy bright  
beames ab- fen- ted,

So far, dear life, dear life, from your bright

5  
beams ab - sen - - - ted,\_\_\_\_\_

### Vertical line as a baroque articulation mark

This short vertical line placed above the note is commonly used in baroque music. Its meaning can vary, but generally indicates notes that should be played with more “weight”. The following example demonstrates how to achieve such a notation.

```

upline =
\tweak stencil
#(lambda (grob)
  (grob-interpret-markup grob #{ \markup \draw-line #'(0 . 1) #}))
\stopped

\relative c' {
  a'4^\upline a( c d')_\upline
}

```

## 18 World music

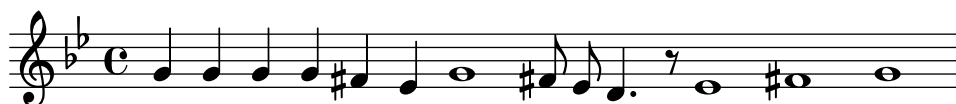
See also Section “World music” in *Notation Reference*.

### Arabic improvisation

For improvisations or *taqasim* which are temporarily free, the time signature can be omitted and `\cadenzaOn` can be used. Adjusting the accidental style might be required, since the absence of bar lines causes 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
}
```



### Makam example

Makam is a type of melody from Turkey using 1/9-tone microtonal alterations.

Consult the initialization file `ly/makam.ly` for details of pitch names and alterations.

```
\include "makam.ly"

\relative c' {
  \set Staff.keyAlterations = #`((6 . ,(- KOMA)) (3 . ,BAKIYE))
  c4 cc db fk
  gbm4 gfc gfb efk
  fk4 db cc c
}
```

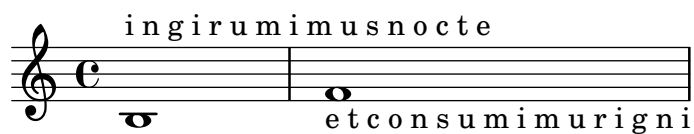


### Printing text from right to left

It is possible to print text from right to left in a markup object, as demonstrated here.

```
{
  b1~\markup {
    \line { i n g i r u m i m u s n o c t e }
  }
  f'~\markup {
    \override #'(text-direction . -1)
    \line { i n g i r u m i m u s n o c t e }
  }
}
```





## Turkish Makam example

This template uses the start of a well-known Turkish *Saz Semai* that is familiar in the repertoire in order to illustrate some of the elements of Turkish music notation.

```

\set-default-paper-size "a6" 'landscape)

```

```

\include "turkish-makam.ly"

```

```

\header {
  title = "Hüseyini Saz Semaisi"
  composer = "Lavtacı Andon"
  tagline = ##f
}

```

```

\relative {
  \set Staff.extraNatural = ##f
  \set Staff.autoBeaming = ##f

```

```

  \key a \huseyni
  \time 10/8

```

```

  a'4 g'16[ fb] e8.[ d16] d[ c d e] c[ d c8] bfc |
  a16[ bfc a8] bfc c16[ d c8] d16[ e d8] e4 fb8 |
  d4 a'8 a16[ g fb e] fb8[ g] a8.[ b16] a16[ g] |
  g4 g16[ fb] fb8.[ e16] e[ g fb e] e4 r8 |

```

```

}

```

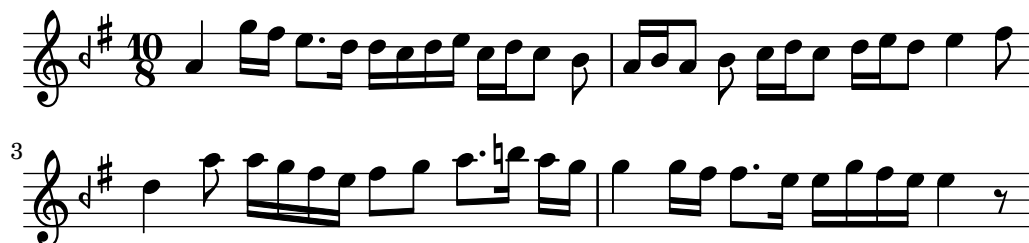
```

\layout {
  indent = 0
}

```

## Hüseyini Saz Semaisi

Lavtacı Andon





## Other collections

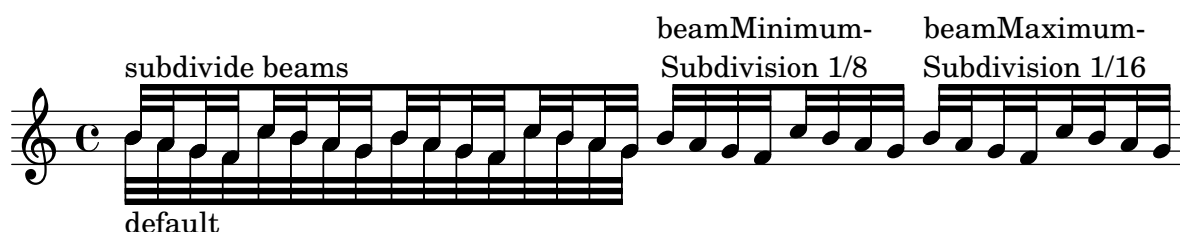


## 19 Automatic notation

### Automatic beam subdivisions

Beams can be subdivided automatically. By setting the property `subdivideBeams`, beams are subdivided whenever possible. The intervals and depth of subdivision can be limited with properties `beamMinimumSubdivision` and `beamMaximumSubdivision`, respectively.

```
\new Staff {
  \relative c'' {
    <<
    {
      \voiceOne
      \set subdivideBeams = ##t
      b32["subdivide beams" a g f c' b a g
      b32 a g f c' b a g]
    }
    \new Voice {
      \voiceTwo
      b32_"default"[ a g f c' b a g
      b32 a g f c' b a g]
    }
    >>
    \oneVoice
    \once \set beamMinimumSubdivision = #1/8
    b32^\markup \center-column { "beamMinimum-"
                                "Subdivision 1/8" } [ a g f c' b a g]
    \once \set beamMaximumSubdivision = #1/16
    b32^\markup \center-column { "beamMaximum-"
                                "Subdivision 1/16" } [ a g f c' b a g]
  }
}
```



### Forcing rehearsal marks to start from a given letter or number

This snippet demonstrates how to obtain automatic ordered rehearsal marks, but from the letter or number desired.

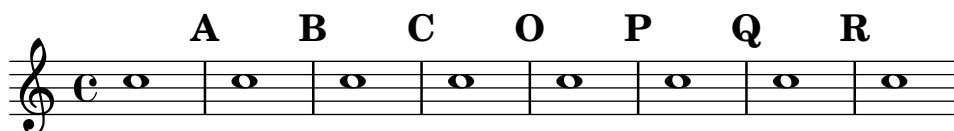
```
\relative c'' {
  \override Score.RehearsalMark.Y-offset = #3.5

  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1 \mark #14
  c1 \mark \default
}
```

```

c1 \mark \default
c1 \mark \default
c1
}

```



## Generating whole scores (also book parts) in Scheme without using the parser

A LilyPond score internally is just a Scheme expression, generated by the LilyPond parser. Using Scheme, one can also automatically generate a score without an input file. If you have the music expression in Scheme, a score can be generated by simply calling

```
(scorify-music music)
```

on your music. This generates a score object, for which you can then set a custom layout block with

```

(let* ((layout (ly:output-def-clone $defaultlayout)))
  ; modify the layout here, then assign it:
  (ly:score-add-output-def! score layout))

```

Finally, all you have to do it to pass this score to LilyPond for typesetting. This snippet defines functions (`add-score score`), (`add-text text`), and (`add-music music`) to pass a complete score, some markup, or some music to LilyPond for typesetting.

This snippet also works for typesetting scores inside a `\book {...}` block as well as top-level scores. To achieve this, each score scheduled for typesetting is appended to the list of top-level scores, and the top-level book handler (which is a Scheme function called to process a book once a `\book{...}` block is closed) is modified to insert all collected scores so far to the book.

Note: For technical reasons, only the first `\book` is shown, as the other `\book` commands create additional output files.

```

#(define-public (add-score score)
  (ly:parser-define! 'toplevel-scores
    (cons score (ly:parser-lookup 'toplevel-scores))))

#(define-public (add-text text)
  (add-score (list text)))

#(define-public (add-music music)
  (collect-music-aux (lambda (score)
    (add-score score))
    music))

#(define-public (toplevel-book-handler book)
  (map (lambda (score)
    (ly:book-add-score! book score))
    (reverse! (ly:parser-lookup 'toplevel-scores)))
  (ly:parser-define! 'toplevel-scores (list))
  (print-book-with-defaults book))

#(define-public (book-score-handler book score)

```

```

    (add-score score))

#(define-public (book-text-handler book text)
  (add-text text))

#(define-public (book-music-handler book music)
  (add-music music))

% Some example code to show how to use these functions. Each call to
% ``oneNoteScore`` constructs a global markup followed by a single
% staff with a single quarter note. The pitch of this note is taken
% from the variable ``pitch``; the start value 0 corresponds to pitch C.
% After emitting the score, variable ``pitch`` gets increased by 1.
%
% ``oneNoteScore`` calls Scheme function ``add-one-note-score`` to do all
% the work.

#(define add-one-note-score #f)
#(let ((pitch 0))
  (set! add-one-note-score
    (lambda ()
      (let* ((music
        (make-music
          'EventChord
          'elements (list (make-music
            'NoteEvent
            'duration (ly:make-duration 2 0 1/1)
            'pitch (ly:make-pitch 0 pitch 0))))))
        (score (scorify-music music))
        (layout (ly:output-def-clone $defaultlayout))
        (note-name (case pitch
          ((0) "do")
          ((1) "ré")
          ((2) "mi")
          ((3) "fa")
          ((4) "sol")
          ((5) "la")
          ((6) "si")
          (else "huh"))
          (title (markup #:large #:line
            ("Score with a" note-name))))
        (ly:score-add-output-def! score layout)
        (add-text title)
        (add-score score))
        (set! pitch (modulo (1+ pitch) 7))))))

oneNoteScore =
#(define-void-function () ()
  (add-one-note-score))

\book {

```

```

\oneNoteScore

\paper { tagline = ##f }
}

\book {
  \oneNoteScore
  \oneNoteScore

  \paper { tagline = ##f }
}

% Top-level scores are also handled correctly.
\oneNoteScore
\oneNoteScore

\paper { tagline = ##f }

```

Score with a do



## Preventing extra naturals from being automatically added

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

```

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

```



## Preventing natural signs from being printed when the key signature changes

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

```

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

```



```

\key g \minor
a4 bes c d
}

```



## Vocal ensemble template with automatic piano reduction

This template adds an automatic piano reduction to the standard SATB vocal score demonstrated in snippet “Vocal ensemble template”. It demonstrates one of the strengths of LilyPond – you can use a music definition more than once. If any changes are made to the vocal notes (say, `tenorMusic`), then the changes also apply to the piano reduction.

```

\paper {
  top-system-spacing.basic-distance = 10
  score-system-spacing.basic-distance = 20
  system-system-spacing.basic-distance = 20
  last-bottom-spacing.basic-distance = 10
}

```

```

global = {
  \key c \major
  \time 4/4
}

```

```

sopMusic = \relative {
  c''4 c c8[( b)] c4
}

```

```

sopWords = \lyricmode {
  hi hi hi hi
}

```

```

altoMusic = \relative {
  e'4 f d e
}

```

```

altoWords = \lyricmode {
  ha ha ha ha
}

```

```

tenorMusic = \relative {
  g4 a f g
}

```

```

tenorWords = \lyricmode {
  hu hu hu hu
}

```

```

bassMusic = \relative {
  c4 c g c
}

```

```

bassWords = \lyricmode {
  ho ho ho ho
}

```

```

}

\score {
  <<
    \new ChoirStaff <<
      \new Lyrics = "sopranos" \with {
        % This is needed for lyrics above a staff
        \override VerticalAxisGroup.staff-affinity = #DOWN
      }
      \new Staff = "women" <<
        \new Voice = "sopranos" { \voiceOne << \global \sopMusic >> }
        \new Voice = "altos" { \voiceTwo << \global \altoMusic >> }
      >>
      \new Lyrics = "altos"

      \new Lyrics = "tenors" \with {
        % This is needed for lyrics above a staff
        \override VerticalAxisGroup.staff-affinity = #DOWN
      }
      \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" \sopWords
      \context Lyrics = "altos" \lyricsto "altos" \altoWords
      \context Lyrics = "tenors" \lyricsto "tenors" \tenorWords
      \context Lyrics = "basses" \lyricsto "basses" \bassWords
    >>

    \new PianoStaff <<
      \new Staff <<
        \set Staff.printPartCombineTexts = ##f
        \partCombine
        << \global \sopMusic >>
        << \global \altoMusic >>
      >>
      \new Staff <<
        \clef bass
        \set Staff.printPartCombineTexts = ##f
        \partCombine
        << \global \tenorMusic >>
        << \global \bassMusic >>
      >>
    >>
  >>
}

```

A musical score for a four-part vocal setting, likely for SATB voices, in common time (C). The score consists of three systems of staves. The first system has two staves: the top staff is in treble clef and the bottom staff is in bass clef. The second system also has two staves: the top staff is in treble clef and the bottom staff is in bass clef. The third system has two staves: the top staff is in treble clef and the bottom staff is in bass clef. The lyrics are: "hi hi hi hi" on the first staff, "ha ha ha ha" on the second staff, "hu hu hu hu" on the third staff, and "ho ho ho ho" on the fourth staff. The melody is simple, using whole and half notes, with a final half note in the first measure of each system.

hi hi hi hi

ha ha ha ha

hu hu hu hu

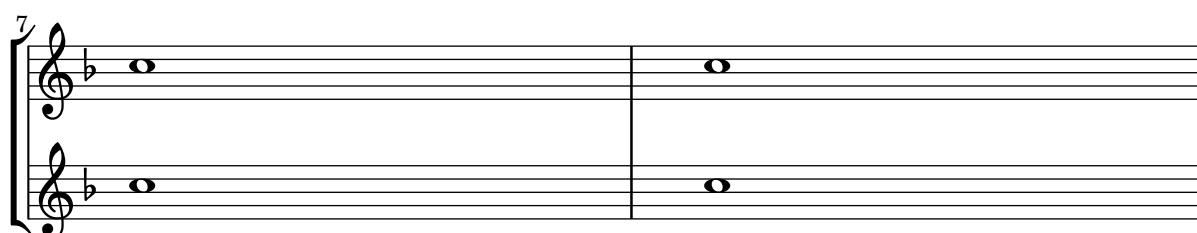
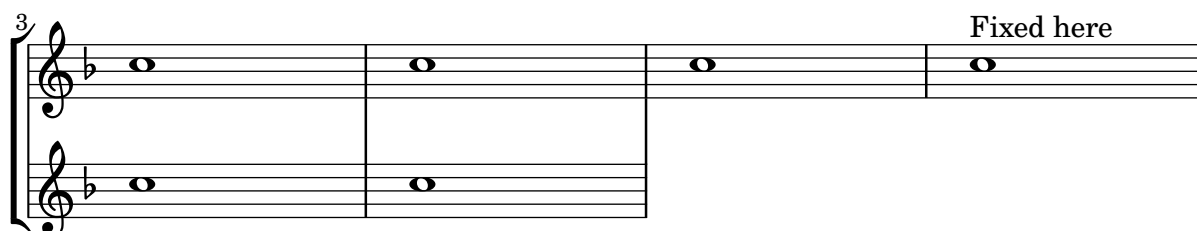
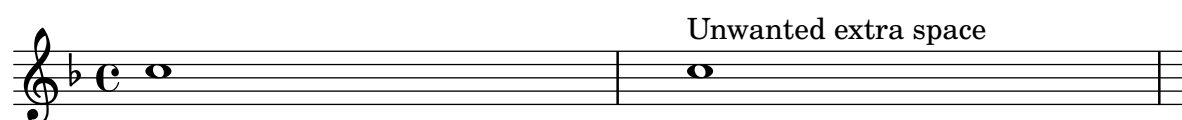
ho ho ho ho

## 20 Breaks

### Adding an extra staff at a line break

When adding a new staff at a line break, some extra space is unfortunately added at the end of the line before the break (to fit in a key signature change, which will never be printed anyway). The workaround is to set the `explicitKeySignatureVisibility` property of the `Staff` grob as is shown in the example.

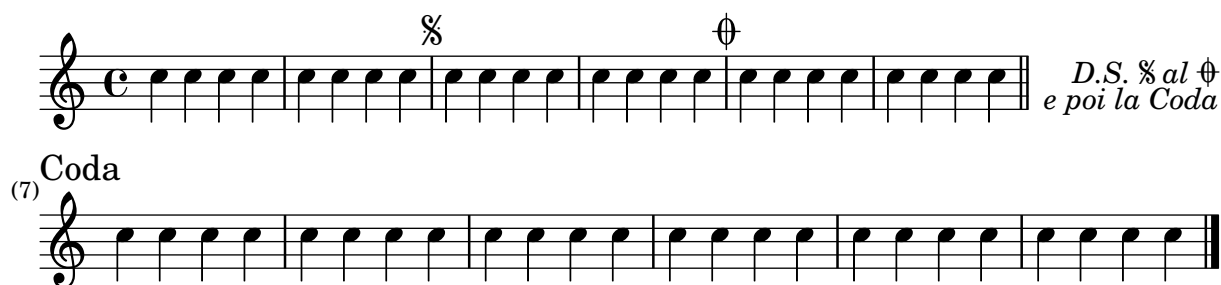
```
\score {
  \new StaffGroup \relative c'' {
    \new Staff
    \key f \major
    c1 c^"Unwanted extra space" \break
    << { c1 | c }
    \new Staff {
      \key f \major
      \once \omit Staff.TimeSignature
      c1 | c
    }
  }
  >>
  c1 | c^"Fixed here" \break
  << { c1 | c }
  \new Staff {
    \once \set Staff.explicitKeySignatureVisibility =
      #end-of-line-invisible
    \key f \major
    \once \omit Staff.TimeSignature
    c1 | c
  }
  >>
}
}
```



## Positioning segno and coda (with line break)

If you want to place an exiting segno sign and add text like “D.S. al Coda” next to it where usually the staff lines are you can use this snippet. The coda will resume in a new line. There is a variation documented in this snippet, where the coda will remain on the same line.

```
\relative c' {
  c4 c c c | c c c c |
  \repeat segno 2 {
    c4 c c c | c c c c |
    \alternative {
      \volta 1 {
        c4 c c c | c c c c |
        % If you don't use \break at Coda, use \noBreak here
        % and after \bar "" below.
        \noBreak
        \section % double bar line
        \cadenzaOn % pause bar count
        \stopStaff % remove staff lines
        % Increasing the unfold counter will expand the staff-free space
        \repeat unfold 4 {
          s1
          \bar ""
        }
        % Place JumpScript where the staff would normally be.
        \once \override Score.JumpScript.outside-staff-priority = ##f
        \once \override Score.JumpScript.Y-offset = 0
        \startStaff % resume bar count
        \cadenzaOff % show staff lines again
      }
    }
  }
}
\sectionLabel "Coda"
% Show Coda on a new line
\break
\repeat unfold 6 { c4 c c c }
\fine
}
```



## Removing the first empty line

To remove the first empty staff from a score, set the `remove-first` property of the `VerticalAxisGroup` grob to `#t`. 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:
    % \override VerticalAxisGroup.remove-first = ##t
  }
}
\new StaffGroup <<
  \new Staff \relative c' {
    e4 f g a \break
    c1
  }
  \new Staff {
    % To use the setting globally, comment this line,
    % uncomment the line in the \layout block above
    \override Staff.VerticalAxisGroup.remove-first = ##t
    R1 \break
    R
  }
}>>
```

```
\new StaffGroup <<
  \new Staff \relative c' {
    e4 f g a \break
    c1
  }
  \new Staff {
    R1 \break
    R
  }
}>>
```

## 21 Connecting notes

### Adding beams, slurs, ties, etc., when using tuplet and non-tuplet rhythms

LilyPond primarily uses postfix syntax for inputting parentheses, brackets, etc., which might feel unintuitive for novices.

For example, when entering a manual beam, the left square bracket has to be placed *after* the starting note and its duration, not before. Similarly, the right square bracket should directly follow the note which is to be at the end of the requested beaming, even if this note happens to be inside a tuplet section.

This snippet demonstrates how to combine manual beaming, manual slurs, ties, and phrasing slurs with tuplet sections (enclosed within curly braces).

```
{
  r16[ g16 \tuplet 3/2 { r16 e'8] }
  g16( a \tuplet 3/2 { b d' e' ) }
  g8[( a \tuplet 3/2 { b d' ) e']\ ( ~ }
  \time 2/4
  \tuplet 5/4 { e'32 a b d' e' } a'4.\)
}
```



### Automatic beam subdivisions

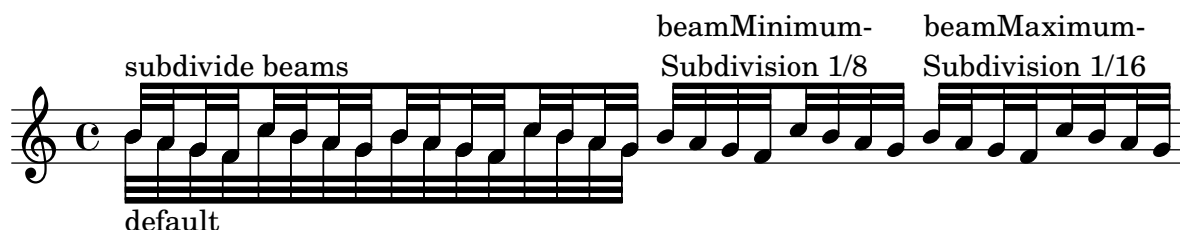
Beams can be subdivided automatically. By setting the property `subdivideBeams`, beams are subdivided whenever possible. The intervals and depth of subdivision can be limited with properties `beamMinimumSubdivision` and `beamMaximumSubdivision`, respectively.

```
\new Staff {
  \relative c' {
    <<
    {
      \voiceOne
      \set subdivideBeams = ##t
      b32["subdivide beams" a g f c' b a g
      b32 a g f c' b a g]
    }
    \new Voice {
      \voiceTwo
      b32_"default"[ a g f c' b a g
      b32 a g f c' b a g]
    }
    >>
    \oneVoice
    \once \set beamMinimumSubdivision = #1/8
    b32~\markup \center-column { "beamMinimum-"
      "Subdivision 1/8" } [ a g f c' b a g]
    \once \set beamMaximumSubdivision = #1/16
```

```

b32^\markup \center-column { "beamMaximum-"
                                "Subdivision 1/16" } [ a g f c' b a g]
}

```



## Changing the appearance of a slur from solid to dotted or dashed

The appearance of slurs may be changed from solid to dotted or dashed.

```

\relative c' {
  c4( d e c)
  \slurDotted
  c4( d e c)
  \slurSolid
  c4( d e c)
  \slurDashed
  c4( d e c)
  \slurSolid
  c4( d e c)
}

```



## 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), `if-no-beam` (only print a bracket if there is no beam) or `#f` (never print a bracket). The latter is in fact equivalent to omitting the `TupletBracket` object altogether from the printed output.

```

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

```

```

\new Voice {
  \relative c' {
    \override Score.TextMark.non-musical = ##f
    \textMark "default" \music
    \override TupletBracket.bracket-visibility = #'if-no-beam

```



```

\textMark \markup \typewriter "'if-no-beam" \music
\override TupletBracket.bracket-visibility = ##t
\textMark \markup \typewriter "#t" \music
\override TupletBracket.bracket-visibility = ##f
\textMark \markup \typewriter "#f" \music
\omit TupletBracket
\textMark \markup \typewriter "omit" \music
}
}

```



## Creating slurs across voices

In some situations it is necessary to create slurs between notes from different voices. The solution is to add invisible notes to one of the voices, using `\hideNotes`.

This example is measure 235 of the Ciaccona from Bach's second partita for solo violin, BWV 1004.

```

\relative c' {
  <<
  {
    d16( a') s a s a[ s a] s a[ s a]
  }
  \\\
  {
    \slurUp
    bes,16[ s e](
    \hideNotes a)
    \unHideNotes f[(
    \hideNotes a)
    \unHideNotes fis](
    \hideNotes a)
    \unHideNotes g[(
    \hideNotes a)
    \unHideNotes gis](
    \hideNotes a)
  }
  >>
}

```



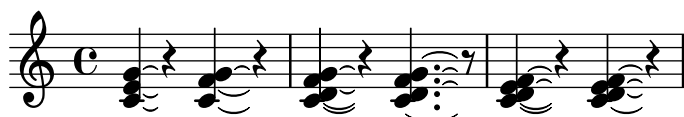
## Laissez vibrer ties

Laissez vibrer ties have a fixed size. Their positioning can be tuned using the `tie-configuration` property.

See also snippet “Longer laissez vibrer ties”.

```
\relative c' {
  <c e g>4\laissezVibrer r <c f g>\laissezVibrer r
  <c d f g>4\laissezVibrer r <c d f g>4.\laissezVibrer r8

  <c d e f>4\laissezVibrer r
  \override LaissezVibrerTieColumn.tie-configuration
    = #'((-7 . ,DOWN)
          (-5 . ,DOWN)
          (-3 . ,UP)
          (-1 . ,UP))
  <c d e f>4\laissezVibrer r
}
```



## Manually controlling beam positions

Beam positions may be controlled manually, by overriding the `positions` setting of the `Beam` grob.

```
\relative c' {
  \time 2/4
  % from upper staff-line (position 2) to center (position 0)
  \override Beam.positions = #'(2 . 0)
  c8 c
  % from center to one above center (position 1)
  \override Beam.positions = #'(0 . 1)
  c8 c
}
```



## 22 Contexts and engravers

See also Section “Changing defaults” in *Notation Reference* and Section “Contexts and engravers” in *Learning Manual*.

### Adding ambitus per voice

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

```
\new Staff <<
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c'' {
    \override Ambitus.X-offset = 2.0
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c' {
    \voiceTwo
    es4 f g as
    b1
  }
}>>
```

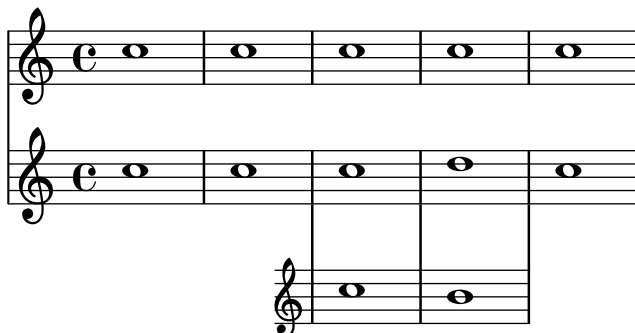


### Adding an extra staff

An extra staff can be added (possibly temporarily) after the start of a piece.

```
\score {
  <<
    \new Staff \relative c'' {
      c1 | c | c | c | c
    }
    \new StaffGroup \relative c'' {
      \new Staff {
        c1 | c
      } <<
      { c1 | d }
      \new Staff {
        \once \omit Staff.TimeSignature
        c1 | b
      }
    }
  >>
  c1
}
}
```

```
>>
}
```



## Adding an extra staff at a line break

When adding a new staff at a line break, some extra space is unfortunately added at the end of the line before the break (to fit in a key signature change, which will never be printed anyway). The workaround is to set the `explicitKeySignatureVisibility` property of the `Staff` grob as is shown in the example.

```
\score {
  \new StaffGroup \relative c'' {
    \new Staff
    \key f \major
    c1 c^"Unwanted extra space" \break
    << { c1 | c }
    \new Staff {
      \key f \major
      \once \omit Staff.TimeSignature
      c1 | c
    }
  }
  >>
  c1 | c^"Fixed here" \break
  << { c1 | c }
  \new Staff {
    \once \set Staff.explicitKeySignatureVisibility =
      #end-of-line-invisible
    \key f \major
    \once \omit Staff.TimeSignature
    c1 | c
  }
  >>
}
}
```



## 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 = #'(-1 . 3)
  \consists "Bar_engraver"
}
```

```
\chordmode {
  f1:maj7 f:7 bes:7
}
```

F<sup>Δ</sup> | F<sup>7</sup> | B<sup>b7</sup> |

## Ambitus after key signature

By default, ambitus are positioned at the left of the clef. The `\ambitusAfter` function allows for changing this placement. Syntax is `\ambitusAfter grob-interface`; see Graphical Object Interfaces (<http://lilypond.org/doc/v2.24/Documentation/internals/graphical-object-interfaces>) for a list of possible values for *grob-interface*.

A common use case is printing the ambitus between key signature and time signature.

```
\new Staff \with {
  \consists Ambitus_engraver
} \relative {
  \ambitusAfter key-signature
  \key d \major
  es'8 g bes cis d2
}
```

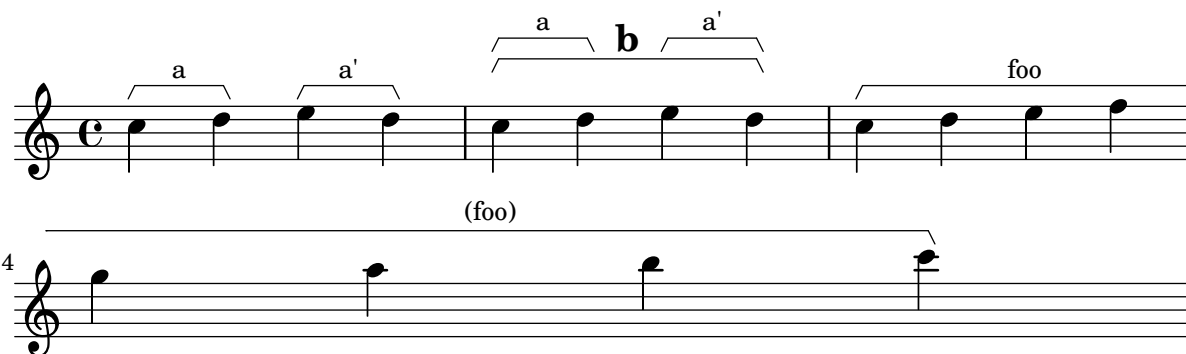
## Analysis brackets with labels

Text markup may be added to analysis brackets using the `text` property of the `HorizontalBracketText` grob. Adding different texts to brackets beginning at the same time requires the `\tweak` command.

Bracket text gets parenthesized after a line break.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
    \override HorizontalBracket.direction = #UP
  }
}

{
  \once\override HorizontalBracketText.text = "a"
  c''\startGroup d''\stopGroup
  \once\override HorizontalBracketText.text = "a'"
  e''\startGroup d''\stopGroup |
  c''-\tweak HorizontalBracketText.text
    \markup \bold \huge "b" \startGroup
    -\tweak HorizontalBracketText.text "a" \startGroup
    d''\stopGroup
    e''-\tweak HorizontalBracketText.text "a'" \startGroup
    d''\stopGroup\stopGroup |
  c''-\tweak HorizontalBracketText.text foo \startGroup
  d'' e'' f'' | \break
  g'' a'' b'' c'''\stopGroup
}
```



## 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.

The context property `suspendMelodyDecisions` may be used to turn off this behavior locally.

```
\relative c'' {
  \time 3/4
  a8 b g f b g |
  \set suspendMelodyDecisions = ##t
  a b g f b g |
  \unset suspendMelodyDecisions
  c b d c b c |
}
```

```
\layout {
```

```

\context {
  \Voice
  \consists "Melody_engraver"
  \autoBeamOff
}
}

```



## Changing MIDI output to one channel per voice

When outputting MIDI, the default behavior is for each staff to represent one MIDI channel, with all the voices on a staff amalgamated. This minimizes the risk of running out of MIDI channels, since there are only 16 available per track.

However, by moving the `Staff_performer` to the `Voice` context, each voice on a staff can have its own MIDI channel, as is demonstrated by the following example: despite being on the same staff, two MIDI channels are created, each with a different `midiInstrument`.

```

\score {
  \new Staff <<
    \new Voice \relative c''' {
      \set midiInstrument = "flute"
      \voiceOne
      \key g \major
      \time 2/2
      r2 g-"Flute" ~
      g fis ~
      fis4 g8 fis e2 ~
      e4 d8 cis d2
    }
    \new Voice \relative c'' {
      \set midiInstrument = "clarinet"
      \voiceTwo
      b1-"Clarinet"
      a2. b8 a
      g2. fis8 e
      fis2 r
    }
  >>
  \layout { }
  \midi {
    \context {
      \Staff
      \remove "Staff_performer"
    }
    \context {
      \Voice
      \consists "Staff_performer"
    }
    \tempo 2 = 72
  }
}

```

}



## Changing time signatures inside a polymetric section using `\scaleDurations`

### *Flexible polymeric with unaligned measures*

To support explicit creation of independently measured contexts, remove the `Timing_translator` from `Score` context and define a `TimingStaffGroup` context that has `Timing_translator`. This makes `Timing` an alias for `TimingStaffGroup`, targeting `\time` commands to the enclosing `TimingStaffGroup`.

Unlike LilyPond's built-in `\enablePerStaffTiming` command, this approach requires the explicit creation of `TimingStaffGroup` contexts; in exchange, it allows creating multiple `Staff` contexts that jointly follow the measure defined in their enclosing `TimingStaffGroup`.

### *Locally scaled time signatures*

Use the unscalable `\time` command to establish a measure of the desired length in `Timing`, a.k.a. `TimingStaffGroup`. In this snippet, all staves below `TimingStaffGroup` use a scaled time signature, so any time signature with the desired measure length is as good as any other. If there were an enclosed context that did not use a scaled time signature, the choice of time signature to set in `Timing` would matter in that context.

Use the `\polymetric \time` command to set scalable metric properties in contexts below `Timing`, and use the `\scaleDurations` command to scale both the local meter and the notes to fit the measure.

```
\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \accepts TimingStaffGroup
  }
  \context {
    \StaffGroup
    \name TimingStaffGroup
    \alias StaffGroup
    \consists "Timing_translator"
  }
}

<<
\new TimingStaffGroup <<
  \new Staff {
    \scaleDurations 8/5 {
      \time 6/5 % to set measure length in Timing
      \context Staff \polymetric \time 6/8
      b8 b b b b b
      \time 4/5 % to set measure length in Timing
```



```

        \context Staff \polymetric \time 2/4
        b4 b
    }
}
>>
\new TimingStaffGroup <<
  \new Staff {
    \clef bass
    \time 2/4
    c2 d e f
  }
>>
>>

```



## 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 }
  >>
}

```



## Creating blank staves

To create blank staves, generate empty measures then remove the `Bar_number_engraver` from the `Score` context, and the `Time_signature_engraver`, `Clef_engraver` and `Bar_engraver` from the `Staff` context.

```

#(set-global-staff-size 10) % for the documentation
% #(set-global-staff-size 20) % for letter and A4

\book {
  \score {

```

```

{ \repeat unfold 12 { s1 \break } }

\layout {
  indent = 0
  \context {
    \Staff
    \remove "Time_signature_engraver"
    \remove "Clef_engraver"
    \remove "Bar_engraver"
  }
  \context {
    \Score
    \remove "Bar_number_engraver"
  }
}

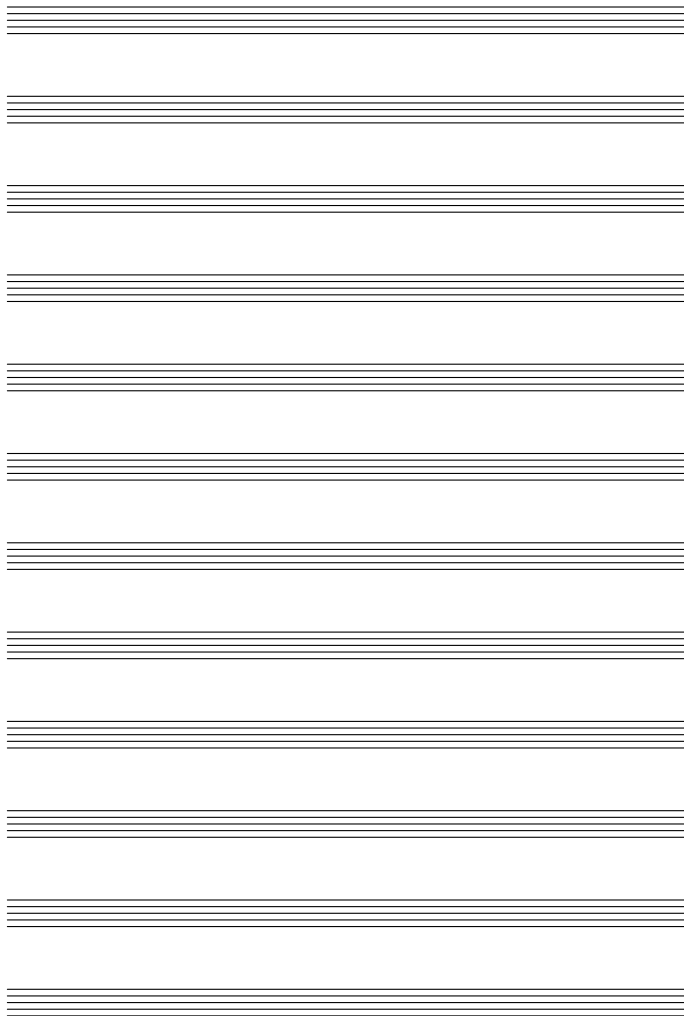
% for the documentation
\paper {
  #(set-paper-size "a6")
  ragged-last-bottom = ##f
  line-width = 90\mm
  left-margin = 7.5\mm
  bottom-margin = 5\mm
  top-margin = 5\mm
  tagline = ##f
}

% uncomment these lines for "letter" size
%{
\paper {
  #(set-paper-size "letter")
  ragged-last-bottom = ##f
  line-width = 7.5\in
  left-margin = 0.5\in
  bottom-margin = 0.25\in
  top-margin = 0.25\in
  tagline = ##f
}
%}

% uncomment these lines for "A4" size
%{
\paper {
  #(set-paper-size "a4")
  ragged-last-bottom = ##f
  line-width = 180\mm
  left-margin = 15\mm
  bottom-margin = 10\mm
  top-margin = 10\mm
  tagline = ##f
}
}

```

```
%}
}
```



## Creating cross-staff arpeggios in other contexts

Cross-staff *arpeggios* can be created in contexts other than `GrandStaff` and its derived siblings (`PianoStaff`, `ChoirStaff`, and `StaffGroup`) if the `Span_arpeggio_engraver` is included in the `Score` context.

```
<<
\new PianoStaff <<
  \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
  }
>>

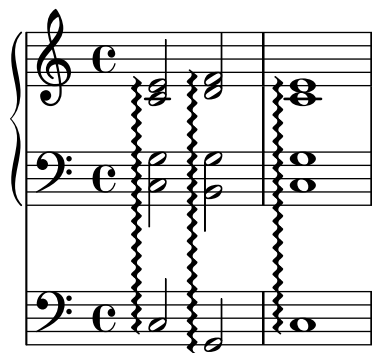
\new Staff \relative c {
```

```

\set Score.connectArpeggios = ##t
\clef bass
c2\arpeggio g\arpeggio
c1\arpeggio
}
>>

\layout {
  \context {
    \Score
    \consists "Span_arpeggio_engraver"
  }
}

```



## Creating custom key signatures

LilyPond supports custom key signatures. In this example, print for D minor and D major with an extended range of shown flats.

```

\new Staff \with {
  \override StaffSymbol.line-count = #8
  \override KeySignature.flat-positions = #'((-7 . 6))
  \override KeyCancellation.flat-positions = #'((-7 . 6))
  \override KeySignature.sharp-positions = #'((-6 . 7))
  \override KeyCancellation.sharp-positions = #'((-6 . 7))

  \override Clef.stencil =
    #(lambda (grob)
      (grob-interpret-markup grob
        #{ \markup\combine
          \musicglyph "clefs.C"
          \translate #'(-3 . -2)
          \musicglyph "clefs.F"
        }
      #})))
  clefPosition = #3
  middleCPosition = #3
  middleCClefPosition = #3
}

{
  \key d\minor f bes, f bes, |
  \key d\major fis b, fis b, |
}

```

}



## Cross-staff stems

This snippet shows how to use `Span_stem_engraver` and `\crossStaff` to connect stems across staves automatically.

The stem lengths need not be specified, as the variable distance between noteheads and staves is calculated automatically. However, it is important that `\crossStaff` is applied to the correct voice or staff (i.e., on the opposite side of where a beam is or would be positioned) to get the desired effect.

```
\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
    \voiceTwo
    % Down to lower staff
    \crossStaff { e'8 e'8 } e'4 |
  }
}
```

```
\new Staff {
  \clef bass
  \voiceOne
  % Up to upper staff
  \crossStaff { <e g>4 e, g16 a8. c8 } d |
  g8 f g4 \voiceTwo g8 g g4 |
}
>>
```



## Defining an engraver in Scheme: ambitus engraver

This example demonstrates how the `ambitus` engraver may be defined on the user side, with a Scheme engraver. This is basically a rewrite in Scheme of the code from `lily/ambitus-engraver.cc`.

```
#(use-modules (oop goops))
```

```

%%%
%%% Grob utilities
%%%
%%% These are literal rewrites of some C++ methods used by the ambitus
%%% engraver.

#(define (ly:separation-item::add-conditional-item grob grob-item)
  "Add GROB-ITEM to the array of conditional elements of GROB.

This is a rewrite of function `Separation_item::add_conditional_item` from
file `lily/separation-item.cc`."
  (ly:pointer-group-interface::add-grob
    grob 'conditional-elements grob-item))

#(define (ly:accidental-placement::accidental-pitch accidental-grob)
  "Get the pitch from the grob cause of ACCIDENTAL-GROB.

This is a rewrite of function `accidental_pitch` from file
`lily/accidental-placement.cc`."
  (ly:event-property (ly:grob-property
    (ly:grob-parent accidental-grob Y) 'cause)
    'pitch))

#(define (ly:accidental-placement::add-accidental grob accidental-grob)
  "Add ACCIDENTAL-GROB to the list of accidentals grobs of GROB.
  ACCIDENTAL-GROB is an `Accidental` grob; GROB is an `AccidentalPlacement`
  grob.

This is a rewrite of function `Accidental_placement::add_accidental` from
file `lily/accidental-placement.cc`."
  (let ((pitch (ly:accidental-placement::accidental-pitch
    accidental-grob)))
    (set! (ly:grob-parent accidental-grob X) grob)
    (let* ((accidentals (ly:grob-object grob 'accidental-grobs))
      (handle (assq (ly:pitch-notename pitch) accidentals))
      (entry (if handle (cdr handle) '()))))
      (set! (ly:grob-object grob 'accidental-grobs)
        (assq-set! accidentals
          (ly:pitch-notename pitch)
          (cons accidental-grob entry))))))

%%%
%%% Ambitus data structures.
%%%

%%% The <ambitus> class holds the various grobs that are created to
%%% print an ambitus:
%%%
%%% - `ambitus-group`: the grob that groups all the components of an
%%%   ambitus (`Ambitus` grob);
%%% - `ambitus-line`: the vertical line between the upper and lower

```

```

%%% ambitus notes (`AmbitusLine` grob);
%%% - `ambitus-up-note` and `ambitus-down-note`: the note head and
%%% accidental for the lower and upper note of the ambitus (see
%%% `` class below).
%%%
%%% The other slots define the key and clef context of the engraver:
%%%
%%% - `start-c0`: position of middle c at the beginning of the piece.
%%% It is used to place the ambitus notes according to their pitch;
%%% - `start-key-sig`: the key signature at the beginning of the
%%% piece. It is used to determine whether accidentals shall be
%%% printed next to ambitus notes.

#(define-class <ambitus> ()
  (ambitus-group #:accessor ambitus-group)
  (ambitus-line #:accessor ambitus-line)
  (ambitus-up-note #:getter ambitus-up-note
    #:init-form (make <ambitus-note>))
  (ambitus-down-note #:getter ambitus-down-note
    #:init-form (make <ambitus-note>))
  (start-c0 #:accessor ambitus-start-c0
    #:init-value #f)
  (start-key-sig #:accessor ambitus-start-key-sig
    #:init-value '()))

%%% Accessor for the lower and upper note data of an ambitus.
#(define-method (ambitus-note (ambitus <ambitus>) direction)
  "Return lower or upper note of AMBITUS depending on DIRECTION."
  (if (= direction UP)
      (ambitus-up-note ambitus)
      (ambitus-down-note ambitus)))

%%% The `` class holds the grobs that are specific to
%%% ambitus (lower and upper) notes:
%%%
%%% - `head`: an `AmbitusNoteHead` grob;
%%% - `accidental`: an `AmbitusAccidental` grob, to be possibly
%%% printed next to the ambitus note head.
%%%
%%% Moreover,
%%%
%%% - `pitch` is the absolute pitch of the note;
%%% - `cause` is the note event that causes this ambitus note, i.e.,
%%% the lower or upper note of the considered music sequence.

#(define-class <ambitus-note> ()
  (head #:accessor ambitus-note-head
    #:init-value #f)
  (accidental #:accessor ambitus-note-accidental
    #:init-value #f)
  (cause #:accessor ambitus-note-cause
    #:init-value #f))

```

```
(pitch #:accessor ambitus-note-pitch
      #:init-value #f))

%%%
%%% Ambitus engraving logic.
%%%
%%% This is rewrite of the code from file `lily/ambitus-engraver.cc`.
```

```
#(define (make-ambitus translator)
  "Build an ambitus object: initialize all the grobs and their
relations.
```

The ``Ambitus`` grob contains all other grobs:

```
Ambitus
|- AmbitusLine
|- AmbitusNoteHead    for upper note
|- AmbitusAccidental  for upper note
|- AmbitusNoteHead    for lower note
|- AmbitusAccidental  for lower note
```

The parent of an accidental is the corresponding note head, and the accidental is set as the ``accidental-grob`` property of the note head so that is printed by the function that prints notes."

```
;; Make the ambitus object.
(let ((ambitus (make <ambitus>)))
  ;; Build the `Ambitus` grob, which will contain all other grobs.
  (set! (ambitus-group ambitus)
        (ly:engraver-make-grob translator 'Ambitus '()))
  ;; Build the `AmbitusLine` grob (the line between lower and upper
  ;; note).
  (set! (ambitus-line ambitus)
        (ly:engraver-make-grob translator 'AmbitusLine '()))
  ;; Build the upper and lower `AmbitusNoteHead` and
  ;; `AmbitusAccidental`.
  (for-each
    (lambda (direction)
      (let ((head (ly:engraver-make-grob translator
                                             'AmbitusNoteHead '()))
            (accidental (ly:engraver-make-grob translator
                                                  'AmbitusAccidental '()))
            (group (ambitus-group ambitus)))
        ;; The parent of the `AmbitusAccidental` grob is the
        ;; `AmbitusNoteHead` grob.
        (set! (ly:grob-parent accidental Y) head)
        ;; The `AmbitusAccidental` grob is set as the
        ;; `accidental-grob` object of `AmbitusNoteHead`. This is
        ;; later used by the function that prints notes.
        (set! (ly:grob-object head 'accidental-grob) accidental)
        ;; Both the note head and the accidental grobs are added to
        ;; the main ambitus grob.
        (ly:axis-group-interface::add-element group head))
    (list 'upper 'lower)))
```



```

(ly:axis-group-interface::add-element group accidental)
;; The note head and the accidental grobs are added to the
;; ambitus object.
(set! (ambitus-note-head (ambitus-note ambitus direction))
      head)
(set! (ambitus-note-accidental (ambitus-note ambitus direction))
      accidental)))
(list DOWN UP))

;; The parent of the ambitus line is the lower ambitus note head.
(set! (ly:grob-parent (ambitus-line ambitus) X)
      (ambitus-note-head (ambitus-note ambitus DOWN)))
;; The ambitus line is added to the ambitus main grob.
(ly:axis-group-interface::add-element (ambitus-group ambitus)
                                       (ambitus-line ambitus))
ambitus))

#(define-method (initialize-ambitus-state
                 (ambitus <ambitus>) translator)
  "Initialize the state of AMBITUS by getting the starting position of
  middle C and key signature from TRANSLATOR's context."
  (if (not (ambitus-start-c0 ambitus))
      (begin
        (set! (ambitus-start-c0 ambitus)
              (ly:context-property (ly:translator-context translator)
                                   'middleCPosition 0))
        (set! (ambitus-start-key-sig ambitus)
              (ly:context-property (ly:translator-context translator)
                                   'keyAlterations)))))

#(define-method (update-ambitus-notes (ambitus <ambitus>) note-grob)
  "Update upper and lower ambitus pitches of AMBITUS using NOTE-GROB."
  ;; Get the event that caused the `note-grob` creation and check
  ;; that it is a `note-event`.
  (let ((note-event (ly:grob-property note-grob 'cause)))
    (if (ly:in-event-class? note-event 'note-event)
        ;; Get the pitch from the note event.
        (let ((pitch (ly:event-property note-event 'pitch)))
          ;; If this pitch is lower than the current ambitus' lower
          ;; note pitch (or it has not been initialized yet), then
          ;; this pitch is the new ambitus' lower pitch. The same is
          ;; done for the upper pitch (but in the opposite
          ;; direction).
          (for-each
           (lambda (direction pitch-compare)
             (if (or (not (ambitus-note-pitch
                           (ambitus-note ambitus direction)))
                     (pitch-compare
                      pitch (ambitus-note-pitch
                             (ambitus-note ambitus direction)))))
               (begin
                 (set! (ambitus-note-pitch
                       (ambitus-note ambitus direction)
                       pitch)
                       (ambitus-note-pitch
                        (ambitus-note ambitus direction)
                        (pitch-compare
                         pitch (ambitus-note-pitch
                               (ambitus-note ambitus direction)
                               direction)))))))
           (list (list DOWN UP) (list UP DOWN))))))

```

[illegible]

```

        (assoc (ly:pitch-notename pitch)
              (ambitus-start-key-sig ambitus))))
      (sig-alter (if handle (cdr handle) 0)))
    (cond
      ((= (ly:pitch-alteration pitch) sig-alter)
       ;; The note alteration is in the key signature
       ;; => it does not have to be printed.
       (ly:grob-suicide! (ambitus-note-accidental
                        (ambitus-note ambitus direction)))
       (set! (ly:grob-object (ambitus-note-head
                            (ambitus-note ambitus direction))
                          'accidental-grob)
             '()))
      (else
       ;; Otherwise the accidental shall be printed.
       (set! (ly:grob-property
            (ambitus-note-accidental
              (ambitus-note ambitus direction)) 'alteration)
             (ly:pitch-alteration pitch))))
      ;; Add the `AccidentalPlacement` grob to the conditional
      ;; items of the `AmbitusNoteHead`.
      (ly:separation-item::add-conditional-item
       (ambitus-note-head (ambitus-note ambitus direction))
       accidental-placement)
      ;; Add the `AmbitusAccidental` to the list of the
      ;; `AccidentalPlacement` grob accidentals.
      (ly:accidental-placement::add-accidental
       accidental-placement
       (ambitus-note-accidental (ambitus-note ambitus direction)))
      ;; Add the `AmbitusNoteHead` grob to the `AmbitusLine` grob.
      (ly:pointer-group-interface::add-grob
       (ambitus-line ambitus)
       'note-heads
       (ambitus-note-head (ambitus-note ambitus direction))))
    (list DOWN UP))
  ;; Add the `AccidentalPlacement` grob to the main `Ambitus` grob.
  (ly:axis-group-interface::add-element
   (ambitus-group ambitus) accidental-placement))
;; No lower and upper pitches => nothing to print.
(begin
  (for-each
   (lambda (direction)
     (ly:grob-suicide! (ambitus-note-accidental
                      (ambitus-note ambitus direction)))
     (ly:grob-suicide! (ambitus-note-head
                      (ambitus-note ambitus direction))))
   (list DOWN UP))
  (ly:grob-suicide! ambitus-line))))

```

```

%%%
%%% Ambitus engraver definition.
%%%

```

```

#(define ambitus-engraver
  (lambda (context)
    (let ((ambitus #f))
      ;; When music is processed, make the ambitus object if not
      ;; already built.
      (make-engraver
        ((process-music translator)
         (if (not ambitus)
             (set! ambitus (make-ambitus translator))))

        ;; Set the ambitus clef and key signature state.
        ((stop-translation-timestep translator)
         (if ambitus
             (initialize-ambitus-state ambitus translator)))

        ;; When a note head grob is built, update the ambitus notes.
        (acknowledgers
         ((note-head-interface engraver grob source-engraver)
          (if ambitus
              (update-ambitus-notes ambitus grob)))))

      ;; Finally, typeset the ambitus according to its upper and
      ;; lower notes (if any).
      ((finalize translator)
       (if ambitus
           (typeset-ambitus ambitus translator))))))

%%%
%%% Example
%%%

```

```

\score {
  \new StaffGroup <<
    \new Staff { c'4 des' e' fis' gis' }
    \new Staff { \clef "bass" c4 des ~ des ees b, }
  >>
  \layout { \context { \Staff \consists #ambitus-engraver } }
}

```



## Displaying a whole GrandStaff system if only one of its staves is alive

In many orchestral scores it is custom to not show staves for instruments that are silent for a while; this is called a ‘Frenched’ score. LilyPond provides this functionality via the `\RemoveEmptyStaves` command.

When they play again it is often preferred to show the staves of *all instruments of such a group*. This can be done by adding the `Keep_alive_together_engraver` to the grouping context (e.g., `GrandStaff` or `StaffGroup`).

In the example below the violins are silent in the second system. Only the first violin plays the last measure in the third system but the staff of the second violin is also displayed.

```
\score {
  <<
    \new Staff = "Staff_flute" \with {
      instrumentName = "Flute"
      shortInstrumentName = "Fl"
    } \relative c' {
      \repeat unfold 3 { c'4 c c c | c c c c | c c c c | \break }
    }

    \new StaffGroup = "StaffGroup_Strings" <<
      \new GrandStaff = "GrandStaff_violins" <<
        \new Staff = "StaffViolinI" \with {
          instrumentName = "Violin I"
          shortInstrumentName = "Vi I"
        } \relative c'' {
          a1 | R1*7 | \repeat unfold 12 a16 a4 |
        }
        \new Staff = "StaffViolinII" \with {
          instrumentName = "Violin II"
          shortInstrumentName = "Vi II"
        } \relative c' {
          e1 | R1*8 |
        }
      >>

      \new Staff = "Staff_cello" \with {
        instrumentName = "Cello"
        shortInstrumentName = "Ce"
      } \relative c {
        \clef bass \repeat unfold 9 { c1 } |
      }
    >>
  >>
}

\layout {
  indent = 3.0\cm
  short-indent = 1.5\cm

  \context {
    \GrandStaff
    \consists Keep_alive_together_engraver
  }
  \context {
    \Staff
    \RemoveEmptyStaves
  }
}
```

}  
}

The image displays three musical score excerpts, each showing four staves: Flute, Violin I, Violin II, and Cello. The excerpts illustrate the incremental addition of engraving elements:

- Excerpt 1:** Shows the initial state where only the staff symbols and clefs are present. The Flute staff has a common time signature 'C'. The Violin I and II staves are grouped with a brace. The Cello staff has a bass clef.
- Excerpt 2:** Shows the addition of note heads. The Flute staff has a sequence of eighth notes. The Violin I and II staves have whole rests. The Cello staff has a whole note.
- Excerpt 3:** Shows the addition of stems, beams, slurs, accents, accidentals, bar lines, time signature, and key signature. The Flute staff has a sequence of eighth notes with stems and beams. The Violin I and II staves have whole rests. The Cello staff has a whole note.

## Engravers one by one

LilyPond handles the various elements necessary to typeset a score with plugins. Each plugin is called an *engraver*. In this example, (some) engravers are switched on one by one, in the following order:

- note heads,
- staff symbol,
- clef,
- stem,
- beams, slurs, accents,
- accidentals, bar lines, time signature, and key signature.

Engravers are grouped. For example, note heads, slurs, beams, etc., form a Voice context. Engravers for key signature, accidentals, bar line, etc., form a Staff context.

```

topVoice = \relative c' {
  \key d \major
  es8([ g] a[ fis])
  b4
  b16[-. b-. b-. cis-.]
  d4->
}

% empty staff and voice contexts
MyStaff = \context {
  \type Engraver_group
  \name Staff
  \accepts Voice
  \defaultchild Voice
}
MyVoice = \context {
  \type Engraver_group
  \name Voice
}

% add note heads
MyVoice = \context {
  \MyVoice
  \consists Note_heads_engraver
}
\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

% add staff
MyStaff = \context {
  \MyStaff
  \consists Staff_symbol_engraver
}
\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

% add clef
MyStaff = \context {
  \MyStaff

```

```

    \consists Clef_engraver
}
\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

% add stems
MyVoice = \context {
  \MyVoice
  \consists Stem_engraver
}
\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

% add beams, slurs, and accents
MyVoice = \context {
  \MyVoice
  \consists Beam_engraver
  \consists Slur_engraver
  \consists Script_engraver
  \consists Rhythmic_column_engraver
}
\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

% add accidentals, bar, time signature, and key signature
MyStaff = \context {
  \MyStaff
  \consists Accidental_engraver
  \consists Bar_engraver
  \consists Time_signature_engraver
  \consists Key_engraver
}
\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

```





## Grid lines: changing their appearance

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

```
\new ChoirStaff <<
  \new Staff {
    \relative c'' {
      \stemUp
      c'4. d8 e8 f g4
    }
  }
  \new Staff {
    \relative c {
      % this moves them up one staff space from the default position
      \override Score.GridLine.extra-offset = #'(0.0 . 1.0)
      \stemDown
      \clef bass
      \once \override Score.GridLine.thickness = 5.0
      c4
      \once \override Score.GridLine.thickness = 1.0
      g'4
      \once \override Score.GridLine.thickness = 3.0
      f4
      \once \override Score.GridLine.thickness = 5.0
      e4
    }
  }
}
>>

\layout {
  \context {
```

```

\Staff
% set up grids
\consists "Grid_point_engraver"
% set the grid interval to one quarter note
gridInterval = #1/4
}
\context {
  \Score
  \consists "Grid_line_span_engraver"
  % this moves them to the right half a staff space
  \override NoteColumn.X-offset = -0.5
}
}

```



## Grid lines: emphasizing rhythms and notes synchronization

Regular vertical lines can be drawn between staves to show note synchronization; however, in case of monophonic music, you may want to make the second staff invisible, and make the lines shorter like in this snippet.

```

\new ChoirStaff {
  \relative c'' <<
  \new Staff {
    \time 12/8
    \stemUp
    c4. d8 e8 f g4 f8 e8. d16 c8
  }
  \new Staff {
    % hides staff and notes so that only the grid lines are visible
    \hideNotes
    \hide Staff.BarLine
    \override Staff.StaffSymbol.line-count = #0
    \hide Staff.TimeSignature
    \hide Staff.Clef

    % dummy notes to force regular note spacing
    \once \override Score.GridLine.thickness = #4.0
    c8 c c
    \once \override Score.GridLine.thickness = #3.0
    c8 c c
    \once \override Score.GridLine.thickness = #4.0
    c8 c c
    \once \override Score.GridLine.thickness = #3.0
    c8 c c
  }
}

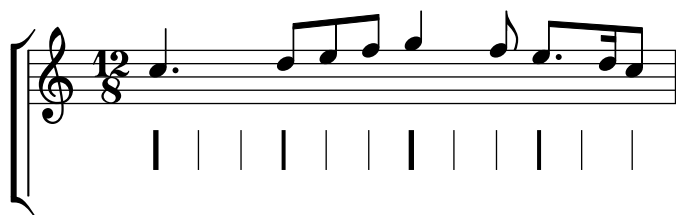
```

```

    }
  >>
}

\layout {
  \context {
    \Score
    \consists "Grid_line_span_engraver"
    % center grid lines horizontally below note heads
    \override NoteColumn.X-offset = #-0.5
  }
  \context {
    \Staff
    \consists "Grid_point_engraver"
    gridInterval = #1/8
    % set line length and positioning:
    % two staff spaces above center line on hidden staff
    % to four spaces below center line on visible staff
    \override GridPoint.Y-extent = #'(2 . -4)
  }
}

```



## Measure counters

This snippet demonstrates the use of the `Measure_counter_engraver` to number groups of successive measures. Any stretch of measures may be numbered, whether consisting of repetitions or not.

The engraver must be added to the appropriate context. Here, a `Staff` context is used; another possibility is a `Dynamics` context.

The counter is begun with `\startMeasureCount` and ended with `\stopMeasureCount`. Numbering will start by default with 1, but this behavior may be modified by overriding the `count-from` property.

When a measure extends across a line break, the number will appear twice, the second time in parentheses.

```

\layout {
  \context {
    \Staff
    \consists #Measure_counter_engraver
  }
}

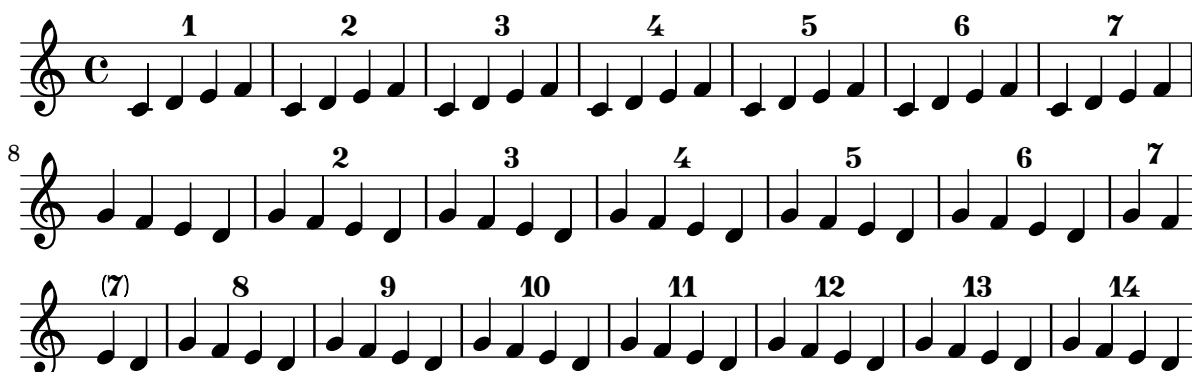
\new Staff {
  \startMeasureCount
  \repeat unfold 7 {

```

```

    c'4 d' e' f'
  }
  \stopMeasureCount
  \bar "||"
  g'4 f' e' d'
  \override Staff.MeasureCounter.count-from = #2
  \startMeasureCount
  \repeat unfold 5 {
    g'4 f' e' d'
  }
  g'4 f'
  \bar ""
  \break
  e'4 d'
  \repeat unfold 7 {
    g'4 f' e' d'
  }
  \stopMeasureCount
}

```



## Measure spanner

Measure spanners are an alternate way to print annotated brackets. As opposed to horizontal brackets, they extend between two bar lines rather than two notes. The text is displayed in the center of the bracket.

```

\layout {
  \context {
    \Staff
    \consists Measure_spanner_engraver
  }
}

```

<<

```

\new Staff \relative c'' {
  \key d \minor
  R1*2
  \tweak text "Answer"
  \startMeasureSpanner
  \tuplet 3/2 8 {
    a16[ b c] d[ c b] c[ d e] f[ e d]
  }
}

```

```

e8 a gis g
fis f e d~ d c b e
\stopMeasureSpanner
}
\new Staff \relative c' {
  \key d \minor
  \tweak text "Subject"
    \tweak direction #DOWN
    \startMeasureSpanner
  \tuplet 3/2 8 {
    d16[ e f] g[ f e] f[ g a] bes[ a g]
  }
  a8 d cis c
  b bes a g~ g f e a
  \stopMeasureSpanner
  \tweak text "Counter-subject"
    \tweak direction #DOWN
    \startMeasureSpanner
  f8 e a r r16 b, c d e fis g e
  a gis a b c fis, b a gis e a4 g8
  \stopMeasureSpanner
}
>>

```

The musical score consists of three staves. The top staff is empty. The middle staff, labeled "Subject", contains a melodic line in D minor, 3/8 time, featuring several triplet figures. The bottom staff, labeled "Counter-subject", contains a counter-melodic line, also in D minor, 3/8 time, with triplet figures. The "Answer" label is positioned above the bottom staff. The score is enclosed in a single system with bar lines between the staves.

## Mensurstriche layout (bar lines between the staves)

*Mensurstriche*, bar lines between but not through staves, can be printed by setting `measureBarType` to `"-span|"` and using a grouping context that allows span bars, such as `StaffGroup`.

```

\layout {
  \context {
    \Staff
    measureBarType = "-span|"
  }
}

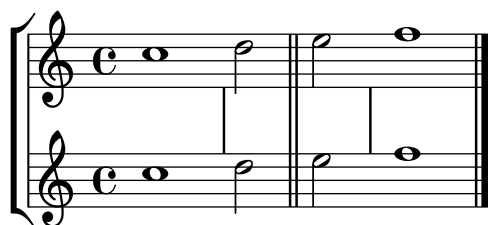
```

```

music = \fixed c'' {
  c1
  d2 \section e2
  f1 \fine
}

\new StaffGroup <<
  \new Staff \music
  \new Staff \music
>>

```



## Nesting staves

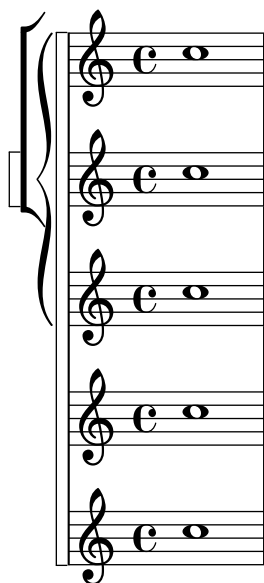
The property `systemStartDelimiterHierarchy` can be used to make more complex nested staff groups. The `systemStartDelimiterHierarchy` property of the `StaffGroup` context takes an alphabetical list of the number of staves produced. Before each staff a system start delimiter can be given. It has to be enclosed in brackets and takes as much staves as the brackets enclose. Elements in the list can be omitted, but the first bracket takes always the complete number of staves. The possibilities are `SystemStartBar`, `SystemStartBracket`, `SystemStartBrace`, and `SystemStartSquare`.

```

\new StaffGroup
\relative c'' <<
  \override StaffGroup.SystemStartSquare.collapse-height = 4
  \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 }
>>

```



## Permitting line breaks within beamed tuplets

These artificial examples show how both manual and automatic line breaks may be permitted within beamed tuplets that can't be rhythmically split in an exact way.

This feature only works with manually beamed tuplets.

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

\relative c'' {
  <>^"manually forced line break"
  a8
  \repeat unfold 5 { \tuplet 3/2 { c8[ b g16 a] } }
  \tuplet 3/2 { c8[ b \break g16 a] }
  \repeat unfold 5 { \tuplet 3/2 { c8[ b g16 a] } }
  c8 \bar "||"
}

\relative c'' {
  <>^"automatic line break"
  \repeat unfold 28 a16
  \tuplet 11/8 { a16[ b c d e f e d c b a] }
  \repeat unfold 28 a16 \bar "||"
}
```

manually forced line break

automatic line break

## Print chord names with same root and different bass as slash and bass note

To print subsequent ChordNames only differing in its bass note as slash and bass note, use the Scheme engraver defined in this snippet. The behaviour may be controlled in detail by the chordChanges context property.

```
#(define Bass_changes_equal_root_engraver
  (lambda (ctx)
    "For sequential `ChordNames` with the same root but a different bass,
    the root markup is dropped: D D/C D/B -> D /C /B.
    The behaviour may be controlled by setting the `chordChanges` context
    property."
    (let ((chord-pitches '())
          (last-chord-pitches '())
          (bass-pitch #f))
      (make-engraver
        ((initialize this-engraver)
         (let ((chord-note-namer (ly:context-property ctx
                                                    'chordNoteNamer)))
           ;; Set 'chordNoteNamer, respect user setting if already done
           (ly:context-set-property! ctx 'chordNoteNamer
                                     (if (procedure? chord-note-namer)
                                         chord-note-namer
                                         note-name->markup))))
        (listeners
         ((note-event this-engraver event)
          (let* ((pitch (ly:event-property event 'pitch))
                 (pitch-name (ly:pitch-notename pitch))
                 (pitch-alt (ly:pitch-alteration pitch))
                 (bass (ly:event-property event 'bass #f))
                 (inversion (ly:event-property event 'inversion #f)))
            ;; Collect notes of the chord
            ;; - to compare inversed chords we need to collect the
```



```

;; bass note as usual member of the chord, whereas an
;; added bass must be treated separate from the usual
;; chord-notes
;; - notes are stored as pairs containing their
;; pitch-name (an integer), i.e. disregarding their
;; octave and their alteration
(cond (bass (set! bass-pitch pitch))
      (inversion
       (set! bass-pitch pitch)
       (set! chord-pitches
              (cons (cons pitch-name pitch-alt)
                    chord-pitches)))
      (else
       (set! chord-pitches
              (cons (cons pitch-name pitch-alt)
                    chord-pitches))))))

(acknowledgers
 ((chord-name-interface this-engraver grob source-engraver)
  (let ((chord-changes (ly:context-property ctx
                                             'chordChanges #f)))
    ;; If subsequent chords are equal apart from their bass,
    ;; reset the 'text-property.
    ;; Equality is done by comparing the sorted lists of this
    ;; chord's elements and the previous chord. Sorting is
    ;; needed because inverted chords may have a different
    ;; order of pitches. `chord-changes` needs to be true.
    (if (and bass-pitch
              chord-changes
              (equal?
               (sort chord-pitches car<)
               (sort last-chord-pitches car<)))
        (ly:grob-set-property!
         grob 'text
         (make-line-markup
          (list
           (ly:context-property ctx 'slashChordSeparator)
           ((ly:context-property ctx 'chordNoteNamer)
            bass-pitch
            (ly:context-property ctx
                                'chordNameLowercaseMinor))))))
        (set! last-chord-pitches chord-pitches)
        (set! chord-pitches '())
        (set! bass-pitch #f))))

((finalize this-engraver)
 (set! last-chord-pitches '()))))

```

```

myChords = \chordmode {
  % \germanChords

```

```

\set chordChanges = ##t
d2:m d:m/cis

d:m/c
\set chordChanges = ##f
d:m/b

e1:7
\set chordChanges = ##t
e
\break

\once \set chordChanges = ##f
e1/f
e2/gis e/+gis e e:m/f d:m d:m/cis d:m/c
\set chordChanges = ##f
d:m/b
}

<<
\new ChordNames
  \with { \consists #Bass_changes_equal_root_engraver }
  \myChords
\new Staff \myChords
>>

```

The image displays two staves of musical notation. The first staff contains six measures with the following chords: Dm, /C#, /C, Dm/B, E<sup>7</sup>, and E. The second staff begins with a measure rest of 5, followed by eight measures with the following chords: E/F, /G#, E, Em/F, Dm, /C#, /C, and Dm/B. Each chord is represented by a vertical stack of notes on a five-line staff, with the chord symbol printed above it.

## Printing marks on every staff

Although rehearsal and text marks are normally only printed above the topmost staff, they may also be printed on every staff.

```

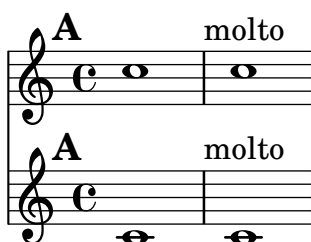
\score {
  <<
    \new Staff { \mark \default c''1 \textMark "molto" c'' }
    \new Staff { \mark \default c'1 \textMark "molto" c' }
  >>
  \layout {
    \context {
      \Score
      \remove Mark_engraver
      \remove Text_mark_engraver
      \remove Staff_collecting_engraver
    }
  }
}

```

```

    }
    \context {
      \Staff
      \consists Mark_engraver
      \consists Text_mark_engraver
      \consists Staff_collecting_engraver
    }
  }
}

```



## Printing music with different time signatures

In the following snippet, two parts have a completely different time signature, yet remain synchronized.

The bar lines can no longer be printed at the Score level; to allow independent bar lines in each part, the `Default_barline_engraver` and `Timing_translator` are moved from the Score context to the Staff context.

If bar numbers are required, the `Bar_number_engraver` should also be moved, since it relies on properties set by the `Timing_translator`; a `\with` block can be used to add bar numbers to the relevant staff.

```

global = {
  \time 3/4 s2.*3 \break
  s2.*3
}

\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Bar_number_engraver"
    \override SpacingSpanner.uniform-stretching = ##t
    \override SpacingSpanner.strict-note-spacing = ##t
    \proportionalNotationDuration = #1/64
  }
  \context {
    \Staff
    \consists "Timing_translator"
  }
  \context {
    \Voice
    \remove "Forbid_line_break_engraver"
    \tupletFullLength = ##t
  }
}

```

```

Bassklarinette = \new Staff \with {
  \consists "Bar_number_engraver"
  barNumberVisibility = #(every-nth-bar-number-visible 2)
  \override BarNumber.break-visibility = #end-of-line-invisible
} <<
\global
{
  \clef treble
  \time 3/8 d''4. |
  \time 3/4 r8 des''2( c''8) |
  \time 7/8 r4. ees''2 ~ |
  \time 2/4 \tupletUp \tuplet 3/2 { ees''4 r4 d''4 ~ } |
  \time 3/8 \tupletUp \tuplet 4/3 { d''4 r4 } |
  \time 2/4 e''2 |
  \time 3/8 es''4. |
  \time 3/4 r8 d''2 r8 |
}
>>

Perkussion = \new StaffGroup <<
\new Staff <<
\global
{
  \clef percussion
  \time 3/4 r4 c'2 ~ |
  c'2. |
  R2. |
  r2 g'4 ~ |
  g'2. ~ |
  g'2. |
}
>>
\new Staff <<
\global {
  \clef percussion
  \time 3/4 R2. |
  g'2. ~ |
  g'2. |
  r4 g'2 ~ |
  g'2 r4 |
  g'2. |
}
>>
>>

\score {
  <<
    \Bassklarinette
    \Perkussion
  >>
}

```

The image displays a musical score with three systems. Each system consists of a single treble staff and a grand staff (two staves). The first system starts with a treble staff containing a dotted quarter note, followed by a measure with a 2/4 time signature, a quarter rest, and a half note. This is followed by a measure with a 7/8 time signature, a quarter rest, and a half note. The system ends with a measure with a 2/4 time signature, a quarter note, and a quarter rest. The grand staff below has a 3/4 time signature and contains a half note, a quarter rest, and a half note. The second system is labeled with a (4) and contains a treble staff with a 3/8 time signature, a quarter note, a quarter rest, and a quarter note. This is followed by a measure with a 2/4 time signature, a quarter note, and a quarter rest. The system ends with a measure with a 3/8 time signature, a quarter note, and a quarter rest. The grand staff below has a 3/4 time signature and contains a half note, a quarter rest, and a half note. The third system is labeled with an 8 and contains a treble staff with a 3/4 time signature, a quarter note, and a quarter rest. The grand staff below has a 3/4 time signature and contains a half note, a quarter rest, and a half note.

## Removing bar numbers from a score

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

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

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

The image shows a musical score with a single treble staff. The staff contains a series of notes: a quarter note, a half note, a quarter note, and a quarter note. The time signature is common time (C). The notes are positioned on the first, second, third, and fourth lines of the staff.



## 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' }
  >> }
}
```



## Using mark lines in a Frenched score

Using `MarkLine` contexts (such as in “Placing rehearsal marks other than above the top staff”) in a Frenched score can be problematic if all the staves between two `MarkLines` are removed in one system. The `Keep_alive_together_engraver` can be used within each `StaffGroup` to keep the `MarkLine` alive only as long as the other staves in the group stay alive.

```
bars = {
  \tempo "Allegro" 4=120
  s1*2
  \repeat unfold 5 { \mark \default s1*2 }
  \bar "||"
  \tempo "Adagio" 4=40
  s1*2
  \repeat unfold 8 { \mark \default s1*2 }
  \bar "|."
}

winds = \repeat unfold 120 { c''4 }
trumpet = { \repeat unfold 8 g'2 R1*16 \repeat unfold 4 g'2 R1*8 }
trombone = { \repeat unfold 4 c'1 R1*8 d'1 R1*17 }
strings = \repeat unfold 240 { c''8 }

#(set-global-staff-size 16)
\paper {
  systems-per-page = 5
  ragged-last-bottom = ##f
  tagline = ##f
}

\layout {
```

```

indent = 16\mm
short-indent = 5\mm
\context {
  \name MarkLine
  \type Engraver_group
  \consists Output_property_engraver
  \consists Axis_group_engraver
  \consists Mark_engraver
  \consists Metronome_mark_engraver
  \consists Staff_collecting_engraver
  \override VerticalAxisGroup.remove-empty = ##t
  \override VerticalAxisGroup.remove-layer = #'any
  \override VerticalAxisGroup.staff-affinity = #DOWN
  \override VerticalAxisGroup.nonstaff-relatedstaff-spacing.padding = 1
  keepAliveInterfaces = #'()
}
\context {
  \Staff
  \override VerticalAxisGroup.remove-empty = ##t
  \override VerticalAxisGroup.remove-layer = ##f
}
\context {
  \StaffGroup
  \accepts MarkLine
  \consists Keep_alive_together_engraver
}
\context {
  \Score
  \remove Mark_engraver
  \remove Metronome_mark_engraver
  \remove Staff_collecting_engraver
  \override BarNumber.Y-offset = #3
}
}

\score {
  <<
  \new StaffGroup = "winds" \with {
    instrumentName = "Winds"
    shortInstrumentName = "W."
  } <<
  \new MarkLine \bars
  \new Staff \winds
  >>
  \new StaffGroup = "brass" <<
  \new MarkLine \bars
  \new Staff = "trumpet" \with {
    instrumentName = "Trumpet"
    shortInstrumentName = "Tp."
  } \trumpet
  \new Staff = "trombone" \with {
    instrumentName = "Trombone"

```

```

        shortInstrumentName = "Tb."
    } \trombone
>>
\new StaffGroup = "strings" \with {
    instrumentName = "Strings"
    shortInstrumentName = "Str."
} <<
    \new MarkLine \bars
    \new Staff = "strings" { \strings }
>>
>>
}

```

Allegro (♩ = 120)

Winds

Trumpet

Trombone

Strings

A B

6

W.

Str.

C D

11

W.

Tb.

Str.

E F

Adagio (♩ = 40)

16

W.

Str.

G H



The image shows a musical score snippet with two systems. The first system (measures 21-25) features three staves: W. (Winds), Tp. (Trumpet), and Str. (Strings). Above the W. staff are labels J, K, and L. The W. staff has a whole note in measure 21, followed by quarter notes in measures 22-25. The Tp. staff has a whole note in measure 21, followed by quarter notes in measures 22-25, and rests in measures 26-29. The Str. staff has a continuous sixteenth-note pattern. The second system (measures 26-30) features two staves: W. (Winds) and Str. (Strings). Above the W. staff are labels M and N. The W. staff has a whole note in measure 26, followed by quarter notes in measures 27-30. The Str. staff has a continuous sixteenth-note pattern.

## Using tags to produce mensural and modern music from the same source

Using tags it is possible to produce both mensural and modern notation from the same music. In this snippet, a function `\menrest` is introduced, allowing mensural rests to be pitched as in the original, but with modern rests in the standard staff position.

Tags can also be used where other differences are needed: for example using “whole measure rests” (`R1`, `R\breve`, etc.) in modern music, but normal rests (`r1`, `r\breve`, etc.) in the mensural version. Converting mensural music to its modern equivalent is usually referred to as *transcription*.

The call `c4.\Be c8 c\Am` is the same as `c4.[ c8 c]`. However, it suppresses warnings if it starts on a note that can’t hold a beam but needs it anyway due to the use of `Completion_heads_engraver`.

[Note that the custos sticks out into the right margin and might be cut off if the LilyPond output gets cropped tightly. The use of `\with-true-dimensions` below avoids this.]

```
\layout {
  line-width = 150\mm
}

menrest = #(define-music-function (note) (ly:music?)
  #{
    \tag #'mens $(make-music 'RestEvent note)
    \tag #'mod $(make-music 'RestEvent note 'pitch '())
  #})

Be = \tag #'mod
  #(begin
    (ly:expect-warning (G_ "stem does not fit in beam"))
    (ly:expect-warning (G_ "beam was started here"))
    (make-span-event 'BeamEvent START))

Am = \tag #'mod ]

MenStyle = {
  \override Score.BarNumber.transparent = ##t
  \override Stem.neutral-direction = #up
```

```

\omit Slur
\omit Beam
}

finalis = \section

Music = \relative c'' {
  \key f \major
  g1 d'2 \menrest bes4 bes a2 \menrest r4 g4 fis4.
  fis8 fis4 fis g e f4.([ g8] a4[ g8 f]
    g2.\Be fis8 e\Am fis2) g\breve \finalis
}

MenLyr = \lyricmode {
  So farre, deere life, deare life,
  from thy bright beames ab- en- ted,
}
ModLyr = \lyricmode {
  So far, dear life, dear life,
  from your bright beams ab -- sen -- ted, --
}

\markup \with-true-dimensions % work around a cropping issue
\score {
  \keepWithTag #'mens {
    <<
    \new PetrucciStaff {
      \new PetrucciVoice = "Cantus" {
        \clef "petrucci-c1" \time 4/4 \MenStyle \Music
      }
    }
    \new Lyrics \lyricsto "Cantus" \MenLyr
  } >>
}
\layout {
  \context {
    \PetrucciVoice
    % No longer necessary starting with version 2.25.23.
    \override Flag.style = #'mensural
  }
}

\markup\vspace #1

\score {
  \keepWithTag #'mod {
    \new ChoirStaff <<
    \new Staff {
      \new Voice = "Sop" \with {
        \remove "Note_heads_engraver"
        \consists "Completion_heads_engraver"

```

```

\remove "Rest_engraver"
\consists "Completion_rest_engraver"
} \shiftDurations 1 0 { \time 2/4 \autoBeamOff \Music }
}
\new Lyrics \lyricsto "Sop" \ModLyr
>>
}
}

```

So farre, deere life, deare life, from thy bright

beames ab- fen- ted,

So far, dear life, dear life, from your bright

beams ab - sen - - - ted,\_\_\_\_

## Vocal ensemble template with verse and refrain

This template creates a score that starts with a solo verse and continues into a refrain for two voices. It also demonstrates the use of spacer rests within the `\global` variable to define meter changes (and other elements common to all parts) throughout the entire score.

```

global = {
  \key g \major

  % verse
  \time 3/4
  s2.*2
  \break

  % refrain
  \time 2/4
  s2*2
  \bar "|."
}

SoloNotes = \relative g' {
  \clef "treble"

  % verse

```

```

    g4 g g |
    b4 b b |

    % refrain
    R2*2 |
}

SoloLyrics = \lyricmode {
    One two three |
    four five six |
}

SopranoNotes = \relative c'' {
    \clef "treble"

    % verse
    R2.*2 |

    % refrain
    c4 c |
    g4 g |
}

SopranoLyrics = \lyricmode {
    la la |
    la la |
}

BassNotes = \relative c {
    \clef "bass"

    % verse
    R2.*2 |

    % refrain
    c4 e |
    d4 d |
}

BassLyrics = \lyricmode {
    dum dum |
    dum dum |
}

\score {
  <<
    \new Voice = "SoloVoice" << \global \SoloNotes >>
    \new Lyrics \lyricsto "SoloVoice" \SoloLyrics

    \new ChoirStaff <<
      \new Voice = "SopranoVoice" << \global \SopranoNotes >>
      \new Lyrics \lyricsto "SopranoVoice" \SopranoLyrics

```

```

\new Voice = "BassVoice" << \global \BassNotes >>
\new Lyrics \lyricsto "BassVoice" \BassLyrics
>>
>>

\layout {
  ragged-right = ##t
  \context { \Staff
    % these lines prevent empty staves from being printed
    \RemoveEmptyStaves
    \override VerticalAxisGroup.remove-first = ##t
  }
}

```

One two three four five six

la la la la

dum dum dum dum

## Volta below chords

By adding the `Volta_engraver` to the relevant staff, volte can be put below chords.

```

\score {
  <<
    \chords { c1 c1 }
    \new Staff \with { \consists "Volta_engraver" }
    {
      \repeat volta 2 { c'1 \alternative { c' } }
    }
  >>
  \layout {
    \context {
      \Score
      \remove "Volta_engraver"
    }
  }
}

```

C C

1.2.

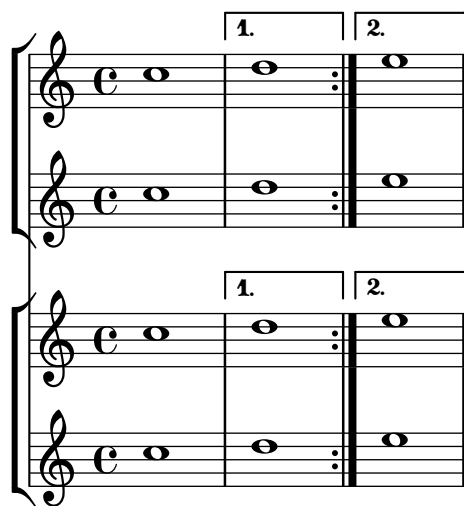
## Volta brackets in multiple staves

By adding the `Volta_engraver` to the relevant staff, volte can be put over staves other than the topmost one in a score.

`\repeat` and related commands should be present in all staves.

```
voltaMusic = \relative c'' {
  \repeat volta 2 {
    c1
    \alternative {
      \volta 1 { d1 }
      \volta 2 { e1 }
    }
  }
}

<<
  \new StaffGroup <<
    \new Staff \voltaMusic
    \new Staff \voltaMusic
  >>
  \new StaffGroup <<
    \new Staff \with { \consists "Volta_engraver" }
      \voltaMusic
    \new Staff \voltaMusic
  >>
>>
```



## 23 Education

### Grid lines: emphasizing rhythms and notes synchronization

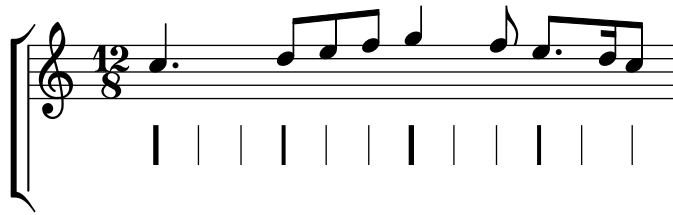
Regular vertical lines can be drawn between staves to show note synchronization; however, in case of monophonic music, you may want to make the second staff invisible, and make the lines shorter like in this snippet.

```
\new ChoirStaff {
  \relative c'' <<
  \new Staff {
    \time 12/8
    \stemUp
    c4. d8 e8 f g4 f8 e8. d16 c8
  }
  \new Staff {
    % hides staff and notes so that only the grid lines are visible
    \hideNotes
    \hide Staff.BarLine
    \override Staff.StaffSymbol.line-count = #0
    \hide Staff.TimeSignature
    \hide Staff.Clef

    % dummy notes to force regular note spacing
    \once \override Score.GridLine.thickness = #4.0
    c8 c c
    \once \override Score.GridLine.thickness = #3.0
    c8 c c
    \once \override Score.GridLine.thickness = #4.0
    c8 c c
    \once \override Score.GridLine.thickness = #3.0
    c8 c c
  }
}
>>
}

\layout {
  \context {
    \Score
    \consists "Grid_line_span_engraver"
    % center grid lines horizontally below note heads
    \override NoteColumn.X-offset = #-0.5
  }
  \context {
    \Staff
    \consists "Grid_point_engraver"
    gridInterval = #1/8
    % set line length and positioning:
    % two staff spaces above center line on hidden staff
    % to four spaces below center line on visible staff
    \override GridPoint.Y-extent = #'(2 . -4)
  }
}
```

}



## 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'
}
```





## 24 Headword

### Ancient headword

Ancient headword

```

#(set-global-staff-size 23)

```

```

\new VaticanaScore {
  <<
    \new VaticanaVoice = "cantus" {
      \clef "vaticana-do3"
      % Verse 1 --- Salve, Regina
      a\melisma \[ a \flexa g \pes a\melismaEnd \] d
      \divisioMinima
      \[ a\melisma \flexa g\melismaEnd \]
      \[ f\melisma \flexa e f \pes g \flexa f\melismaEnd \]
      \[ e\melisma \flexa d\melismaEnd \]
      \divisioMaior
      c d \[d\melisma \flexa c\melismaEnd \] d
      \[ e\melisma \pes f\melismaEnd\] g
      \[d\melisma \pes e \flexa c\melismaEnd \] d
      \finalis
      % Verse 2 --- Vita, dulcédo
      % a\melisma \[ a \flexa g \pes a\melismaEnd \] d
      % \divisioMinima
      % \[ a\melisma \flexa g\melismaEnd \]
      % \[ f\melisma \flexa e f \pes g \flexa f\melismaEnd \]
      % \[ e\melisma \flexa d\melismaEnd \]
      % \divisioMaior
      % c d \[e\melisma \pes f\melismaEnd \] g
      % \[d\melisma \pes e \flexa c\melismaEnd \] d
      % \finalis
      % Verse 3 --- Ad te clamámus
      \[ d\melisma \pes f\melismaEnd\] a g
      \[ g\melisma \flexa f \pes a\melismaEnd\] e
      \divisioMaior
      g f \[ e\melisma \flexa d \pes g\melismaEnd \]
      \divisioMinima
      c d \[ e\melisma \flexa d \pes g\melismaEnd \]
      \[ f\melisma \flexa e\melismaEnd \] d
      \finalis
      % Verse 4 --- Ad te suspirámus
      \[ d\melisma \pes f\melismaEnd \] a c' g
      \[ g\melisma \flexa f \pes g\melismaEnd \] a
      \divisioMaior
      d \[ f\melisma \pes \deminutum g\melismaEnd \] g d
      \[ \virga f\melisma \inclinatum e \inclinatum d\melismaEnd \]
      c \divisioMaior
      d \[ d\melisma \flexa c \pes f\melismaEnd \]
      \[ g\melisma \pes a\melismaEnd \]
      g \[ f\melisma \flexa e\melismaEnd \] g
    }
  }
}

```

```

\[ f\melisma \flexa \deminutum e\melismaEnd \]
\[ d\melisma \flexa c \pes d\melismaEnd \]
\finalis
% Verse 5 --- Eia ergo, Advocáta nostra
f\melisma \[ f \pes g\melismaEnd \] f
\[ g\melisma \pes \deminutum a\melismaEnd \] a
\divisioMinima
c' g \[ \virga a\melisma \inclinatum g \inclinatum f\melismaEnd \] d g a
\divisioMaior
d' d' \[ c'\melisma \flexa b c' \pes d'\melismaEnd \] a
\divisioMinima
d' c' a \[ g\melisma \flexa f \pes a\melismaEnd \] g
\[ d\melisma \pes e\melismaEnd \] f
\[ \virga e\melisma \inclinatum d \inclinatum c\melismaEnd \]
\divisioMaior
\[ c\melisma \pes d\melismaEnd \]
f \[ g\melisma \flexa \deminutum f\melismaEnd \]
\[ d\melisma \flexa c \pes d\melismaEnd \] d
\finalis
% Verse 6 --- Et Jesum
d a, \[ c\melisma \pes d\melismaEnd \]
\divisioMinima
d \[ d\melisma \pes e\melismaEnd \] \[ e\melisma \flexa d d\melismaEnd \]
c g f \[ e\melisma \flexa \deminutum d\melismaEnd \] g
\[ f\melisma \flexa e\melismaEnd \]
\[ d\melisma \flexa c \pes d\melismaEnd \]
\[ d\melisma \pes a \] \virga bes\melismaEnd a
\divisioMinima
\[ \virga a\melisma \inclinatum g \inclinatum f\melismaEnd \] g d f
\[ f\melisma \flexa e\melismaEnd \] \[ d\melisma \flexa c\melismaEnd \]
\divisioMinima
\[ e\melisma \pes f \flexa e\melismaEnd \] d d
\finalis
% Verse 7 ad finem --- O clemens: O pia: O dulcis Virgo María
a\melisma \[ a \flexa g a \quilisma b \pes c'\melismaEnd \]
\[ \virga b\melisma \inclinatum a \inclinatum g\melismaEnd \] a
\finalis
\[ g\melisma \pes a \quilisma b \pes c' \] \[ c' \flexa b\melismaEnd \]
\[ a\melisma \flexa g\melismaEnd \] \[ g\melisma \pes a\melismaEnd \]
\finalis
\[ a\melisma \flexa d \virga f \inclinatum e \inclinatum d \inclinatum c d \]
\divisioMinima
\[ d \pes e f \pes g\melismaEnd \]
\[ g\melisma \flexa \deminutum f\melismaEnd \]
\[ g\melisma \pes a\melismaEnd \]
d c d \[ d\melisma \pes g \flexa f f\melismaEnd \]
\[ e\melisma \flexa d\melismaEnd \]
\finalis
}

```

```

\new VaticanaLyrics \lyricsto "cantus" {
  Sal -- ve, Re -- gí -- na, ma -- ter mi -- se -- ri -- cór -- di -- ae:

```

```

% Vi -- ta, dul -- cé -- do, et spes no -- stra, sal -- ve.
Ad te cla -- má -- mus, éx -- su -- les, fi -- li -- i He -- vae.
Ad te su -- spi -- rá -- mus, ge -- mén -- tes et flen -- tes
  in hac la -- cri -- má -- rum val -- le.
E -- ia er -- go, Ad -- vo -- cá -- ta no -- stra,
  il -- los tu -- os mi -- se -- ri -- cór -- des
  ó -- cu -- los ad nos con -- vér -- te.
Et Je -- sum, be -- ne -- díc -- tum fruc -- tum ven -- tris tu -- i,
  no -- bis post hoc ex -- sí -- li -- um os -- tén -- de.
O cle -- mens:
O pi -- a:
O dul -- cis Vir -- go Ma -- rí -- a.
}
>>
}

```

```

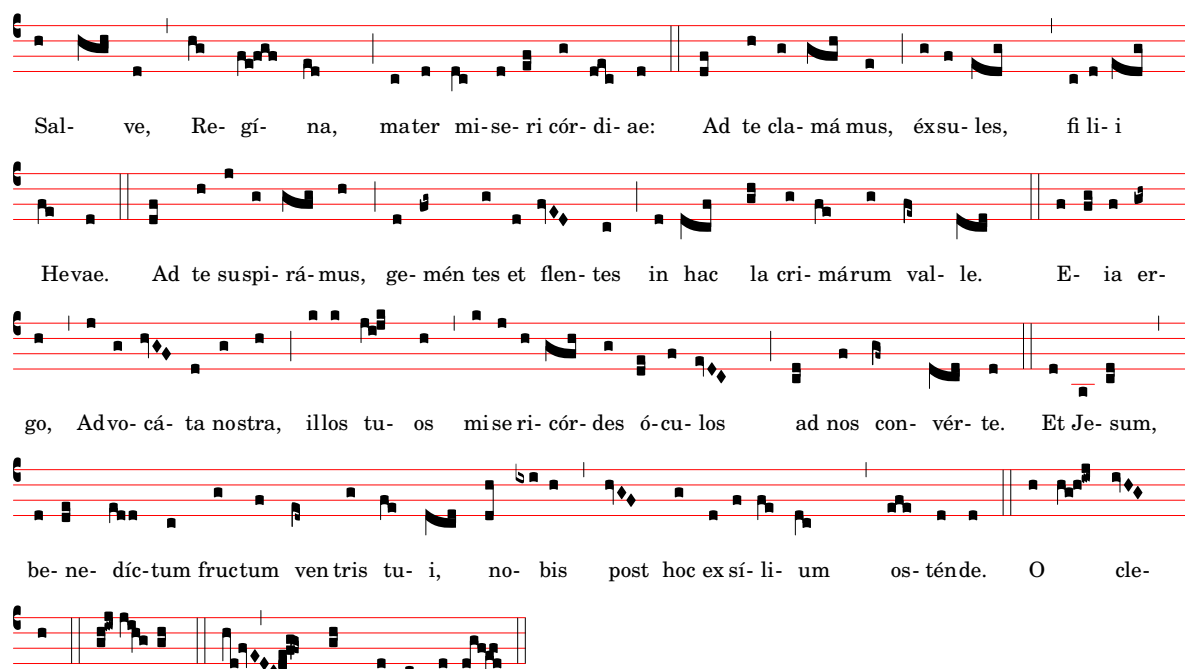
\layout {
  indent = 0
  ragged-last = ##t
}

```

```

\paper { tagline = ##f }

```



Sal- ve, Re- gí- na, mater mi-se- ri cór- di- ae: Ad te cla- má mus, éxsu- les, fi li- i

Hevae. Ad te suspi- rá- mus, ge- mén tes et flen- tes in hac la cri- márum val- le. E- ia er-

go, Advo- cá- ta nostra, illos tu- os mi-se- ri cór- des ó-cu- los ad nos con- vér- te. Et Je- sum,

be- ne- díc- tum fructum ven tris tu- i, no- bis post hoc ex sí- li- um os- tén- de. O cle-

mens: O pi a: O dul cis Virgo Ma rí a.

## Chords headword

Chords headword.

```

theChords = \chordmode {
  \time 2/2
  f1 | c2 f2 | f1 | c2 f2| %\break
  f2 bes2 | f1 | c2:7 f | c1 | \break
}

```

```

verseOne = \lyricmode {
  \set stanza = #"1."
  Fair is the sun - shine,
  Fair - er the moon - light
  And all the stars _ _ _ in heav'n a -- bove;
}

verseTwo = \lyricmode {
  \set stanza = #"2."
  Fair are the mead - ows,
  Fair - er the wood - land,
  Robed in the flow -- ers of bloom -- ing spring;
}

Soprano = {
  \time 2/2
  \key f \major
  \stemUp
  f'2 f'4 f' | g'4 e' f'2 | a'4. a'8 a'4 a' | bes'4 g' a'2 |
  c''2 f''4 d'' | c''2 bes'4 a' | bes'2 a' | g'1 |
}

Alto = {
  \key f \major
  c'2 c'4 c' | d'4 c' c'2 | f'4. f'8 f'4 fis' | g'4 e' f'2 |
  f'2 f'4 f' | f'2 g'4 f' | e'2 f' | e'1 |
}

Tenor = {
  \key f \major
  \stemDown
  a2 a4 a | bes4 g a2 | c'4. c'8 d'4 d' | d'4 c' c'2 |
  a2 d'4 bes | a2 c'4 c' | c'2 c' | c'1 |
}

Bass = {
  \key f \major
  f2 f4 f | bes,4 c f2 | f4. e8 d4 c | bes,4 c f2 |
  f2 bes,4 d | f2 e4 f | g2 f | c1 |
}

\score {
  <<
    \new ChordNames { \theChords }
    \context Staff = upper {
      \context Voice = sop {
        <<
          \Soprano
          \Alto
        >>
      }
    }
  }
}

```

```

    }
  }
  \context Lyrics = "LyrOne" \lyricsto "sop" { \verseOne }
  \context Lyrics = "LyrTwo" \lyricsto "sop" { \verseTwo }
  \context Staff = lower {
    \new Voice {
      \clef bass
      \accidentalStyle modern-cautionary
      <<
        \Tenor
        \Bass
      >>
    }
  }
}
>>

\layout {
  indent = 0
  \context {
    \Score
    \remove "Bar_number_engraver"
  }
  \context {
    \Voice
    \override StanzaNumber.padding = #1.8
  }
}

```

1. Fair is the sun - shine, Fair - er the moon - light  
 2. Fair are the mead - ows, Fair - er the wood - land,

And all the stars in heav'n a - bove;  
 Robed in the flow - ers of bloom - ing spring;

## Editorial headword

Editorial headword.

```

\include "english.ly"

% Beethoven, Op. 31, No. 3
% Piano sonata 18, Movt II, Scherzo
% Measures 9 - 14

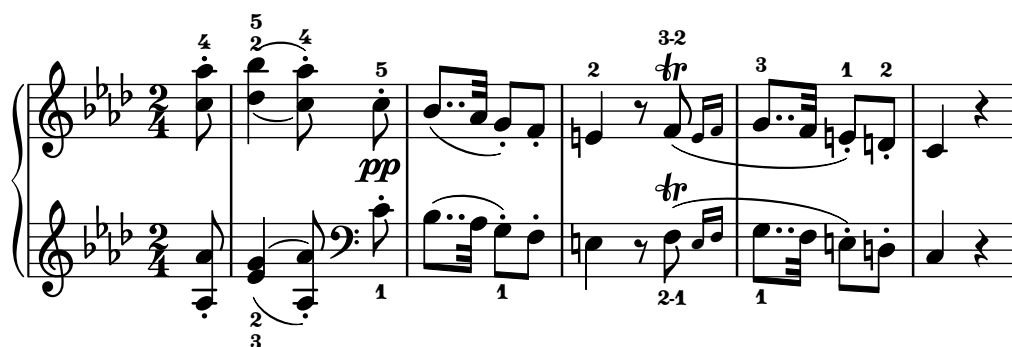
\new PianoStaff <<
  \new Staff = "right hand" {
    \clef treble
    \key af \major
    \time 2/4
    \set Staff.fingeringOrientations = #'(up)
    \set Score.currentBarNumber = #9

    \partial 8 <af''-4 c''>8-. |
    \once \set doubleSlurs = ##t
    <bf''-5 df''-2>4( <af''-4 c''>8-. ) \noBeam c''8-5-. \pp |
    bf'8..( af'32 g'8)-. f'8-. |
    e'4-2 r8
    \once \override Script.script-priority = #-100
    \afterGrace f'8(\trill^\finger "3-2" { e'16[ f'16] } |
    g'8..-3 f'32 e'8-1)-. d'8-2-. |
    c'4 r4 |
  }

  \new Staff = "left hand" {
    \key af \major
    \clef treble
    \override Fingering.direction = #down
    \set Staff.fingeringOrientations = #'(down)

    \partial 8 <af' af>8-.
    \once \set doubleSlurs = ##t
    <g'-2 ef'-3>4( <af' af>8)-. \noBeam \clef bass c'8-1-. |
    bf8..( af32 g8-1)-. f8-. |
    e4 r8 \afterGrace f8(\trill_\finger "2-1" { e16[ f16] } |
    g8..-1 f32 e8)-. d8 -. |
    c4 r4 |
  }
}
>>

```



## Expressive headword

Expressive headword.

```
% L. v. Beethoven, Op. 49 no. 1
% Piano sonata 19 - "Leichte Sonate"
% measures 1 - 12

\include "english.ly"

\new PianoStaff <<
  \new Staff = "right hand" {
    \clef treble
    \key g \major
    \time 6/8

    \partial 2
    \textMark \markup \override #'(baseline-skip . 3)
      \column { RONDO
        \italic Allegro }

    d'8-. d'-. g'-. a'-. |
    b'8[( g')] e'-. e'-. a'-. b'-. |
    c'8[( a')] e'-. d'-. c'-. b'-. |
    a'8-. g'-. a'-. \acciaccatura { g'16[ a'] } bf'8 a'-. g'-. |
    fs'8[( d')] d'-. d'-. g'-. a'-. |
    % 5
    b'8[( g')] e'-. e'-. a'-. b'-. |
    c'8[( a')] e'-. d'-. c'-. b'-. |
    a'8-. g'-. a'-. << { d' g' fs' } \
      { d'4 c'8 } >> |
    <b g'>4-- d'8-. g'-. b'-. d'-. |
    d'8( <c' a'>-. ) <c' a'>-. d'8( <b' g'>-. ) <b' g'>-. |
    % 10
    d'8( <c' a'>-. ) <c' a'>-. d'8( <b' g'>-. ) <b' g'>-. |
    d'8-. <c' a'>-. <b' g'>-. d'-. <c' a'>-. <b' g'>-. |
    <d' c' a'>4\fermata r8 r4 r8 |
  }

  \new Staff = "left hand" {
    \clef bass
    \key g \major
    \time 6/8

    \partial 2 r8 r <d' b>-. <c' a>-. |
    <b g>4 r8 r <e' c'>-. <d' b>-. |
    <c' a>4 r8 r <a fs>-. <b g>-. |
    <c' a>8-. <b d'>-. <e' c'>-. <e' cs'>4.( |
    d'4) r8 r <d' b!>-. <c'! a>-. |
    % 5
    <b g>4 r8 r <e' c'>-. <d' b>-. |
    <c' a>4 r8 r <a fs>-. <b g>-. |
    <c' a>8-. <d' b>-. <e' c'>-. <b d>4 <a d>8-. |
    <g g,>4 \tenuto r8 r4 r8 |
  }
```

```

r8 <d' fs>-. <d' fs>-. r <d' g>-. <d' g>-. |
% 10
r8 <d' fs>-. <d' fs>-. r <d' g>-. <d' g>-. |
r8 <d' fs>-. <d' g>-. r <d' fs>-. <d' g>-. |
<d' fs>4\fermata r8 r4 r8 |
}
>>

```

RONDO  
*Allegro*

## Figured bass headword

Figured bass headword.

```

% Arcangelo Corelli, 12 Sonate da Camera, Op. 2
% Sonata II, Allemanda
% measures 1 - 7

```

```

extendOn = \bassFigureExtendersOn
extendOff = \bassFigureExtendersOff

\score {
  \new StaffGroup <<
    \new GrandStaff <<
      \new Staff = "violinoI" \with { instrumentName = "Violino I." }
      {
        \time 4/4
        \tempo Adagio
        \partial 8 r16 a' |
        a'8.[ d''16 d''8. e''16] cis''8 a'4 a''16 bes'' |
        cis''8 d''16( e'') e''8. d''16 d''4 r8 d''16 e'' |
        f''8 f''4 g''16( f'') e''8 e''4 f''16( e'') |
        d''8. d''16 g''( f'') e''( d'') cis''8 cis''4 cis''16 cis'' |
        d''8 d'' c''8. c''16 c''8( b'4) b'16 b' |
        c''8 c'' bes'8. bes'16 bes'8( a'4) a''16 a'' |
        a''8 g'' g''8. g''16 g''8( f'') r f'' |
      }
    }
  \new Staff = "violinoII" \with { instrumentName = "Violino II." }
}

```



```

{
  \time 4/4
  \partial 8 r16 f' |
  f'8. g'16 g'4 a' r8 d''16 d'' |
  e''8 a' cis''8. d''16 d''4 r8 f''16 g'' |
  a''8 a'' d''8. d''16 g'8 g' c''8. c''16 |
  f'8. f''16 bes''( a'') g''( f'') e''8 e''4 e''16 e'' |
  a'8 fis'' g'' a'' d'' d''4 d''16 d'' |
  g'8 e'' f'' g'' c'' c''4 cis''16 cis'' |
  d''8 d'' e''8. e''16 e''8 a' r d'' |
}
>>

\new Staff = "violone" \with {
  instrumentName = \markup {
    \center-column { Violone,
      "e Cembalo." } } }
{
  \time 4/4
  \clef bass
  \partial 8 r16 d |
  d4 bes, a, f |
  g8 f16 g a8 a, d4 d' ~ |
  d'8 c' b4 c'8 c'16 bes a4 |
  bes8 bes16 a g4 a8 a,4 a16 g |
  fis8 d e fis g8 g,4 g16 f |
  e8 c d e f8 f,4 a,8 |
  b,4 cis d r8 d' |
}

\new FiguredBass \figuremode {
  \set figuredBassAlterationDirection = #RIGHT
  \set figuredBassPlusDirection = #RIGHT
  \override BassFigureAlignment.stacking-dir = #DOWN
  s8 |
  s4 <6> <_+> <6> |
  <6 4\+ 2>8 <6> <_+> s s2 |
  <5>8 <6 4> <6 5>4 s <5>8 <6> |
  s4 <6 5 _-> <_+>2 |
  <6>8 <_+> <6> <6 5> <5 4> \extendOn <5 _!> \extendOff s4 |
  <6>4 <6->8 <6 5-> <5 4-> \extendOn <5 3>4 \extendOff <5 _+>8 |
  <7>8 <6> <5>4 <9 4>8 <8 3> s4 |
}
>>
}

\layout {
  indent = 3\cm
}

```

**Adagio**

Violino I.

Violino II.

Violone,  
e Cembalo.

## Fretted headword

Fretted headword.

% Johann Kaspar Mertz, *Opern-Revue* op. 8

% No. 17 ("Bellini, Norma")

% measures 123 - 133

%%% shortcuts

% fingering orientations

sfol = \set fingeringOrientations = #'(left)

sfor = \set fingeringOrientations = #'(right)

sfod = \set fingeringOrientations = #'(down)

sfou = \set fingeringOrientations = #'(up)

```

% string number orientations
ssnol = \set stringNumberOrientations = #'(left)  %(down right up)
ssnou = \set stringNumberOrientations = #'(up)
ssnod = \set stringNumberOrientations = #'(down)
ssnor = \set stringNumberOrientations = #'(right)

% define fingering offset
FO = #(define-music-function (offsetX offsetY) (number? number?)
#{
  \once \override Voice.Fingering.extra-offset = #(cons offsetX offsetY)
#})

% markups
rit = \markup \bold \italic "rit."
dimin = \markup \italic "dim."
benmarcato = \markup \italic \bold "il canto ben marcato"
pdolce = #(make-dynamic-script
  (markup #:line (#:dynamic "p" #:normal-text #:italic "dol.")))

% triplet
T = \tuplet 3/2 \etc

%%% THE MUSIC %%%

melody = \relative c {
  \voiceOne
  \clef "treble_8"
  \key d \major
  \once \omit Staff.TimeSignature
  \time 4/4

  \sfol
  e,32[ a' c e] e,[ a c e] e,,[ a' c e] e,[ a c e]
    f4\rest <e'-4>4-> | % m. 1
  e,,32[ gis' b e] e,[ gis b e] e,,[ gis' b e] e,[ gis b e]
    f4\rest \FO #0.4 #0.5 <gis-1 e'-4>4 | % m. 2
  d4\rest <b e>-> d4\rest^\rit <b e>4-> | % m. 3
  <gis b e>1 \bar "||" % m. 4

  \tempo \markup \larger \italic "Andantino"
  \key a \minor
  \time 4/4
  R1 | % m. 5
  e'4^\benmarcato e8. d16-4
    d4-4 \T { \sfou \FO #-0.3 #0.6 <c-2>4 b8 } | % m. 6
  \FO #-0.3 #0.3 <a-3>4 \T { c4 b8 } a4 e'8. e16 | % m. 7
  \FO #-0.3 #0.3 <g-4>4 \T { \sfol \FO #0.3 #0.0 <f-1>4 e8 }
    e4 \T { \sfou <d-4>4 c8 } | % m. 8
  b4 \T { d4-4 c8 } \sfou \FO #-1.7 #-1.5 <b-0>4 e | % m. 9
  e4 e8. d16-4 d4 \T { c4 b8 } | % m. 10
  \T { a4 a8 b4 c8 }

```

```

    \sfou \F0 #-0.3 #0.3 <d-4>4~\< \T { e4 <d f>8\! } | % m. 11
}

bass = \relative c {
  \voiceTwo
  \key d \major
  \time 4/4

  e,,8\fp[ e'] e,[ e'] e, \sfol <c''-1> <a'-2> c, | % m. 1
  e,,8\fp[ e'] e,[ e']
    e, \sfod \F0 #0.2 #-0.2 <b''-1> \sfol \F0 #0.3 #0.0 <e-1> b | % m. 2
  e,,8 e' gis e e, e' gis_\dimin e | % m. 3
  e,1 | % m. 4

  \T { a8\p e' a c a e a, e' a c a e } | % m. 5
  \T { a,8\pdolce e' a c a e }
    \T { e,8 \sfou <e'-3> <gis-1> c gis e } | % m. 6
  \T { a,8 <e'-2> a c e, b' a, e' a c a e } | % m. 7
  \T { f,8 f' a \sfol \F0 #0.3 #-0.5 <d-4> a f }
    \T { fis, d' a' d a d, } | % m. 8
  \T { <g,-3>8 d' g d' g, d }
    \T { \sfod <gis,-4> \sfou <e'-2> <gis-1> b gis e } | % m. 9
  \T { a,8 e' a c a e e, e' gis c gis e } | % m. 10
  \T { a,8 e' a b a e f, f' a d a f } | % m. 11
}

\new Staff = "guitar" <<
  \context Voice = "upper" { \melody }
  \context Voice = "lower" { \bass }
>>

\layout {
  \context {
    \Score
    \remove "Bar_number_engraver"
    \override Fingering.staff-padding = #'()
    \omit TupletNumber
    \override TupletBracket.bracket-visibility = ##f
  }
}

\paper {
  system-system-spacing.padding = 3
}

```

*fp*

*fp*

*rit.*

*dim.*

*Andantino*

*p*

*il canto ben marcato*

*p dol.*

## Keyboard headword

Keyboard headword

% M. Ravel, *Sonatine* (1905)

% End of first movement

\include "english.ly"

```
\layout {
  \context {
    \Score
    \remove "Bar_number_engraver"
  }
}
```

```
fermataLong = \markup {
  \override #'(direction . 1)
  \override #'(baseline-skip . 2) {
    \dir-column {
      \fermata
      \serif \italic \center-align long
    }
  }
}
```

```

\new PianoStaff <<
  \set PianoStaff.connectArpeggios = ##t
  \new Staff {
    \time 2/4
    \key fs \major
    <<
      \new Voice {
        \voiceOne
        \textMark \markup {
          \override #'(baseline-skip . 2.4) \column {
            \line \bold { Un peu retenu }
            \line \italic { très expressif } } }
        fs''8( es''16 cs'' as'4) |
        fs''8( es''16 cs'' as'4) |
        fs''8( es''16 cs'' as'8 cs''8) |
      }
      \new Voice {
        \voiceTwo
        gs'8\rest \offset Y-offset #-3 \ppp fs'4( es'8) |
        gs'8\rest fs'4( es'8) |
        gs'8\rest fs'4( es'8) |
      }
    >>
  \clef bass
  \override TextSpanner.bound-details.left.text = "rall."
  \override TextSpanner.bound-details.right.text = "a tempo"
  <b! es'>4(\startTextSpan
  \override Script.stencil =
    #(\lambda(grob)
      (grob-interpret-markup grob fermataLong))
  <ds' as'>8)\fermata \noBeam
  \clef treble
  <as fs'>8^(
    \tweak to-barline ##f
    \tweak after-line-breaking ##f
    \tweak endpoint-alignments #'(-1 . 0) ^\>
    \stopTextSpan | \noBreak
  <gs b cs'>4.\!)
  <as fs'>8^(
    \tweak to-barline ##f
    \tweak after-line-breaking ##f
    \tweak endpoint-alignments #'(-1 . 0) ^\> |
  <gs b cs'>4.\!)
  <<
    \new Voice {
      \voiceOne
      <as fs'>8( |
      \override TextSpanner.bound-details.left.text =
        "ral - - len - - tan - - do"
      \override TextSpanner.bound-details.right.text =
        \markup \larger \upright \bold "Lent"
    }
  >>

```

```

cs'8\<\startTextSpan b16 cs'
  d'8\tweak to-barline ##f \> e'16 fs' |
<as! cs' gs'>4.)\! s8 |
r8 <cs'' as'' cs'''>4\arpeggio
  e''16(\stopTextSpan fs''16 |
\voiceTwo
<as!! cs'' gs''>2) |
}
\new Voice {
  \voiceTwo
  s8 |
  <gs b>4 <fs bs>4 |
  s4. <a bs e'>8^(^> \tweak to-barline ##f _\> |
  <as! cs' gs'>4.)\!
  <a' bs'>8\tweak X-offset #-4 \ppp \tweak to-barline ##f \> |
  s8\!
  \voiceOne
  \ottava 1
  \once \override PianoStaff.Arpeggio.padding = 0.8
  <cs''' as''' cs''''>4. \arpeggio \fermata
  \ottava 0
  \bar "|."
}
>>
}

\new Staff <<
  \key fs \major
  \clef bass
  \new Voice {
    \voiceOne
    ds'4-- cs'4-- |
    ds'4-- cs'4-- |
    ds'4-- cs'4-- |
    r8 \clef treble <b' cs''>8[ \clef bass <es b cs'>8]\fermata
    s8~\tweak Y-offset 3 \pp |
    fs8\rest \clef treble <b' cs''>4-- s8 |
    fs8\rest \clef treble <b' cs''>4-- s8 |
    s2 |
    ds8\rest \clef treble <as' cs''>4 \clef bass s8 |
    s8 \clef treble <as'>4 \arpeggio \clef bass s8 |
    s8 \clef treble <as''>4. \arpeggio \fermata |
  }
  \new Voice {
    \voiceTwo
    ds'8[( <ds bs> cs' <ds as>)] |
    ds'8[( <ds bs> cs' <ds as>)] |
    ds'8[( <ds bs> cs' <ds as>)] |
    \set Staff.pedalSustainStrings = #("P" "" "")
    <cs, gs, ds>4.\sustainOn \fermata
    <fs, cs>8(\sustainOff
      \tweak to-barline ##f

```

```

\new {
  \tweak after-line-breaking ##f _\> |
  <e, b,>4.)\! \clef bass
  <fs, cs>8( \tweak to-barline ##f
    \tweak after-line-breaking ##f _\> |
  <e, b,>4.)\! \clef bass <fs, cs>8( |
  <e, b,>4 <d, a,> |
  <fs,, cs,>4.) <a, e>8( |
  <fs, cs>4.) <a e'>8^( |
  <fs cs'>2) |
}
>>
>>

\paper {
  system-system-spacing.padding = 4
}

```

**Un peu retenu**  
*très expressif*

The musical score is for a piece titled "Un peu retenu très expressif". It is written for piano in 2/4 time with a key signature of three sharps (F#, C#, G#). The score consists of three measures. The right hand (treble clef) plays a series of eighth notes, while the left hand (bass clef) plays a series of quarter notes. The dynamics are marked *ppp* (pianissimo) at the beginning of the first measure.

*rall.* - - - - *long* - - - - *a tempo*

The musical score is for a piece titled "rall. - - - - long - - - - a tempo". It is written for piano in 2/4 time with a key signature of three sharps (F#, C#, G#). The score consists of four measures. The right hand (treble clef) plays a series of eighth notes, while the left hand (bass clef) plays a series of quarter notes. The dynamics are marked *pp* (pianissimo) at the beginning of the first measure. The tempo markings "rall.", "long", and "a tempo" are placed above the first, second, and third measures respectively.





## Pitches headword

Pitches headword.

% L. v. Beethoven

% Piano sonata 21 op. 53 - "à Monsieur Comte de Waldstein"

% first movement, measures 34 - 41

`\include "english.ly"`

`\new PianoStaff <<`

`\new Staff = "right hand" <<`

`\new Voice = "right hand voice 1" {`

`\set Score.currentBarNumber = 34`

`\voiceOne gs''2(^{\markup \italic "dolce e molto legato" fs''4 e'' |`

`ds''2 cs'') |`

`ds''2( e''4 fs'' |`

`<gs'' e''>2 <fs'' ds'') \clef bass |`

`\oneVoice <gs' e' b>2( <fs' ds' a>4 <e' cs' gs> |`

`<ds' bs fs>2 <cs' a e>) |`

`\voiceOne b2\tweak height-limit 7`

`\tweak positions #'(6 . 2) ( cs'4 ds' \clef treble |`

`<e' gs>4) r4 r2 |`

`}`

`\new Voice = "right hand voice 2" {`

`\voiceTwo <e'' b'>2 <ds'' a'>4 <cs'' gs'> |`

`<bs' fs'>2 e' |`

`<b'! a'>2 b'4 <e'' cs'')> |`

`b'2.( a'4) \clef bass | \break`

`s1 |`

`s1 |`

`<gs e>4( <a fs>2.) |`

`s4 r4 r2 |`

`}`

`>>`

`\new Dynamics {`

```

s1*2\p |
s1\tweak style #'none \cresc |
s1\sf\> |

s1*4\p |
}

\new Staff = "left hand" {
  \override Staff.SustainPedalLineSpanner.staff-padding = 6
  <gs' e'>2(\sustainOn <fs' ds' b>4\sustainOff <e' cs'> |
  <ds' bs gs>2 <cs' a>)\sustainOn \clef bass |
  <ds' b! a fs>2_(\sustainOff
    <e' b gs>4 <fs' cs' a>\sustainOn \clef treble |
  << { \voiceOne <gs' e'>2 <fs' ds'> }
    \new Voice { \voiceTwo b1\sustainOff } >> \clef bass |

  \oneVoice <gs e>2( <fs ds b,>4 <e cs> |
  <ds bs, gs,>2 <cs a,>)\sustainOn |
  <b,! b,,!>1(\sustainOff |
  <e e,>4) r4 r2 |
}
>>

```

*dolce e molto legato*

34

38

Red. \*

## Repeats headword

Repeats headword

```

% Beethoven, Op. 57
% Piano sonata 23 -
% (Appassionata, "Dem Grafen Franz von Brunsvik gewidmet")
% Movt II, Andante con moto
% Measures 9 - 16

```

```

\include "english.ly"

\new PianoStaff <<
  \new Staff = RH {
    \clef treble
    \key df \major
    \time 2/4
    \set Score.currentBarNumber = #9

    \bar ".|:-|"
    \repeat volta 2 {
      \change Staff = LH \voiceOne <af ef c>4 <af gf c>8.. <af ef c>32 |
      <af f df>8. <df' af f>16 <c' af gf>8 <df' af f>8 |
      <af ef c>4 <af gf c>8.. <af ef c>32 |
      <af f df>8. \change Staff = RH f'16 f'^( ef' df'8)

      \change Staff = LH \voiceOne <af ef c>4 <af gf c>8.. <af ef c>32 |
      \change Staff = RH <af' df' af>8. <af' f' df'>16
      << { af'16( gf' f'8) } \\\
      { <ef' c'>8 <df' af> } >> |
      ef'4\tweak height-limit 5 ^(
      \change Staff = LH \voiceOne <af ef c> |
      <df' f df>4.) r8
    }
  }

\new Staff = LH {
  \clef bass
  \key df \major
  \time 2/4

  \repeat volta 2 {
    \voiceTwo <af, gf,>4 <af, ef,>8.. <af, gf,>32 |
    <af, f,>8. <af, df,>16 <af, ef,>8 <af, df,> |
    <af, gf,>4 <af, ef,>8.. <af, gf,>32 |
    << { \voiceTwo s4 gf8\tweak positions #'(-0.3 . -0.8) [ f] } \\\
    { \voiceOne s8. \crossStaff {<af f>16 af s af8 } } \\\
    { \voiceTwo <af, f,>8. <af, df,>16
      \once\shiftOn <af, c,>8 <af, df,> } >> |

    <af, gf,>4 <af, ef,>8.. <af, gf,>32 |
    \oneVoice <f f,>8.( <df df,>16 <ef ef,>8 <f f,>) |
    << { \voiceOne \crossStaff <gf bf>4 s |
      s2 } \\\
      { \voiceTwo gf,4 af,( ~ |
        af,16.[ gf,32 f,16. ef,32]) df,8 r8 } >> |
    }
  }
>>

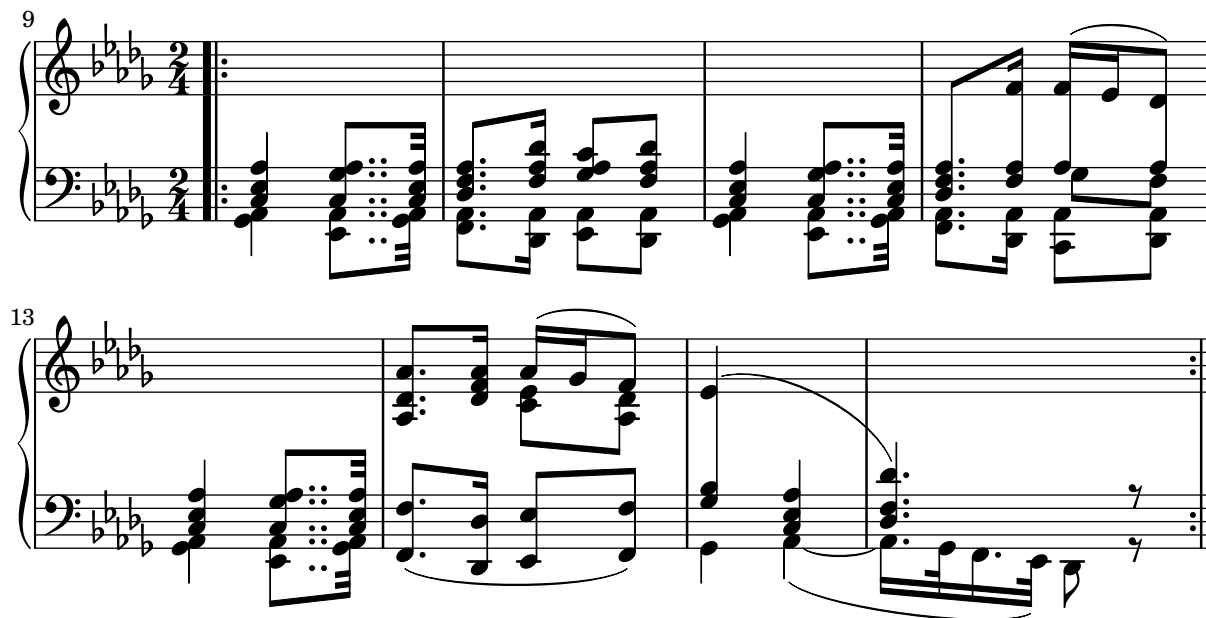
\layout {

```

```

\context {
  \PianoStaff
  \consists "Span_stem_engraver"
}
}

```



## Rhythms headword

Rhythms headword.

```

% Beethoven, Op. 81a
% Piano sonata 26 - (Les Adieux)
% Movt II - Abwesenheit (L'Absence)
% Measures 31 - 34

```

```

\include "english.ly"

```

```

% Circumvent issue #6876: `strict-grace-spacing` ignores
% accidentals of the following main note.

```

```

shiftedGrace =

```

```

#(define-music-function (offset music) (number? ly:music?)
  #{
    \override NoteHead.X-offset = #(- offset 0.85)
    \override Stem.X-offset = #offset
    \grace { $music }
    \revert NoteHead.X-offset
    \revert Stem.X-offset
  })

```

```

\new PianoStaff <<
  \new Staff = "right hand" {
    \clef treble
    \key c \minor
    \time 2/4
    \set Score.currentBarNumber = #31
  }

```

```

<c''' c'>8(^\markup \italic "a tempo" <g' g'>) ~
  <g' g'>8( <a' a'>16 <f' f'>) |
<f' f'>8[( \shiftedGrace #-0.15 { e'32 f' e' d' }
  <e'! e'>16 <f' f'>16)] <g' g'>16-.([ <a' a'>-.)]
  <bf' bf'>32( <b' b'>) <b' b'>( <c''' c'>) |

b''32([ c''' d''' c'''32]) g''8 ~
  g''32[ a''64( g'')] a''64([ g'') bf''( a''64)]
  bf''64([ a'') c'''( b''64)] c'''128[ b' d''' c''' f'''64 f''] |
<f' f'>8[( \shiftedGrace #-0.15 { e'32 f' e' d' }
  <e'! e'>16 <f' f'>)] <g' g'>16-.([ <af'! af'!>-.)]
  <bf' bf'>32( <b' b'>) <b' b'>( <c''' c'>) |
}

\new Dynamics {
  s2\offset Y-offset 1 -\markup \italic "cantabile" |
  s4 s\tweak style #'none \cresc |
  s16.\offset Y-offset 1 \p \offset Y-offset 1 \> s32\! s4. |
  s4 s\tweak style #'none \cresc <>\! |
}

\new Staff = "left hand" {
  \set Staff.beatBase = #1/8
  \set Staff.beatStructure = 1,1,1,1
  \clef bass
  \key c \minor
  \time 2/4
  \repeat unfold 3 { <g e>32 c' <g e> c' } <a f> c' <a f> c' |
  \repeat unfold 2 { <bf g>32 c' <bf g> c' }
    <bf g> c' <a f> c' <g e> c' <g e> c' |

  \repeat unfold 3 { <g e>32 c' <g e> c' } <a f> c' <a f> c' |
  \repeat unfold 2 { <bf g>32 c' <bf g> c' }
    <bf g> c' <af! f> c' <g e> c' <g e> c' |
}
>>

\layout {
  \context {
    \Score
    \override SpacingSpanner.base-shortest-duration =
      \musicLength 1*1/40
    \override SpacingSpanner.strict-grace-spacing = ##t
  }
}

```

31 *a tempo*  
*cantabile* *cresc.*

33 *p*

34 *cresc.*

## Simultaneous headword

Simultaneous headword.

```
\include "english.ly"
```

```
% L. v. Beethoven, Op. 111
```

```
% Piano sonata 32
```

```
% Movt II - Arietta - Adagio molto semplice e cantabile
```

```
% measures 108 - 118
```

```
\layout {
  \context {
    \Score
  }
}
```

```
trillFlat =
\once \override TrillSpanner.bound-details.left.text = \markup {
  \concat {
    \musicglyph "scripts.trill"
    \translate #'(-0.5 . 1.9)
    \fontsize #-7 \flat
  }
}
```

```
\new PianoStaff <<
  \new Staff = "right hand" <<
```

```

\set Score.currentBarNumber = #108
\new Voice = "right hand 1" {
  \clef treble
  \key c \major
  \time 9/16

  \grace s32 s4. s8. |
  s4. \voiceOne a''8[(\p g''16)] |
  g''4.\dim af''8[( g''16)] |
  g''8.[ g''8. g''8.] |
  g''8.[\pp af''8.af''8.] |
  af''8.[ af''8.af''8.] |

  \trillFlat af''4.\startTrillSpan\< ~ af''8. ~ |
  af''4.\> ~ af''8. ~ |
  \oneVoice <af'' d''>8.[\p\cresc a''8. bf''8.] ~ |
  bf''8.[ b''8. c'''8.] ~ \bar "||"
  \key ef \major c'''8.[ cs'''8.] <>\stopTrillSpan <>\!
}

\new Voice = "right hand 2" {
  \override Voice.TrillSpanner.direction = #DOWN
  \grace cs''32 \voiceTwo d''4.\f\startTrillSpan ~ d''8. ~ |
  d''4. ~ d''8. ~ |
  d''8. <>\stopTrillSpan\trillFlat d''4.\startTrillSpan ~ |
  d''4. ~ d''8. ~ |
  d''4. ~ d''8. ~ |
  d''4. ~ d''8. ~ <> \stopTrillSpan |

  \trillFlat d''4.\startTrillSpan ~ d''8. ~ |
  d''4. ~ d''8. ~ |
  \once \override NoteColumn.ignore-collision = ##t
  \hideNotes d''8.\stopTrillSpan s4. |
  s4. s8. |
  s4.
}
>>

\new Staff = "left hand" {
  \clef bass
  \key c \major
  \time 9/16

  \grace s32 r8. r8. <c! c,!>8[(\tweak X-offset #-2 _\f <g, g,,>16)] |
  <g, g,,>4. \clef treble c''8[( b'16)] |
  b'4. c''8[( b'16)] |
  b'8.[ b'8. b'8.] |
  b'8.[ bf'8.] \clef bass <f f,>8[( <bf, bf,,>16)] |
  <bf, bf,,>4. \clef treble f'8[( bf16)] |

  <<
  \new Voice {

```

```

\voiceOne
\override Voice.TrillSpanner.direction = #UP
f'4.~ \startTrillSpan f'8.~ |
f'4.~ f'8.~ |
f'8. <> \stopTrillSpan
}
\new Voice {
\voiceTwo
\override Voice.TrillSpanner.direction = #DOWN
bf8.[ bf8. bf8.] |
bf8.[ bf8. bf8.] |
bf8.
}
>> \oneVoice r8. r8. |
R1*9/16 \clef bass |
\key ef \major r8. r8.
}
>>

```

The musical score is for a piano piece, likely from 'Danse Napolitaine' by Tchaikovsky. It is in 9/16 time and E-flat major. The score consists of two staves. The treble staff has a trill in the right hand, and the bass staff has a trill in the left hand. The trills are marked with 'f' (forte), 'p' (piano), 'dim.' (diminuendo), and 'pp' (pianissimo). The score includes dynamic markings and a crescendo marking 'p cresc.'.

## Staff headword

Staff headword.

```
\include "catalan.ly"
```

```

% Piotr Ilitch Tchaïkovski
% Le Lac des Cygnes, op. 20
% Danse Napolitaine
% arr. Laurence Sardain (Mutopia 2006/12/22)

```

```
\set-global-staff-size 18)
```



```

trompette = \relative do'' {
  \clef treble
  \key mib \major
  \time 2/4

  <>\tweak staff-padding #1.5 ^\markup { \larger \italic Comodo } R2 |
  r8 \once \override TextScript.padding = #2.0
    sib16-.\markup { \dynamic p \italic grazioso } do-.
    mib16( re)-. do-. sib-. |
  re8-. r8 re4->( |
  re8) do16-. re-. mib( re) do-. re-. |
  do8-. r8 sib4-> |
}

tambourin = \drummode {
  \time 2/4
  r8 tamb16 16 8 8 |
  r8 16 16 8 8 |
  r8 8 r8 8 |
  r8 16 16 8 8 |
  r8 8 r8 8 |
}

upper = \relative do' {
  \clef treble
  \key mib \major
  \time 2/4

  r8\p <sol sib mib>16-. q-. q8-. q-. |
  r8 <sol sib mib>16-. q-. q8-. q-. |
  r8 <lab sib re>16-. q-. q8-. q-. |
  r8 <lab sib re>16-. q-. q8-. q-. |
  r8 <sol sib mib>16-. q-. q8-. q-. |
}

lower = \relative do {
  \clef bass
  \key mib \major
  \time 2/4

  mib4-. r4 |
  sib-. r |
  fa'-. r |
  sib,-. r |
  mib4-. r4 |
}

\score {
  <<
    \context Staff = "trumpet" \with {
      instrumentName = \markup {

```

```

    "Trumpet" \concat { B \teeny \raise #0.4 \flat } }
} \transpose sib do' \trompette
\context RhythmicStaff = "tambourin" \with {
  instrumentName = "Tambourine"
} \tambourin
\context PianoStaff = "prima" \with {
  instrumentName = "Piano"
} <<
  \context Staff = "uppera" \upper
  \context Staff = "lowera" \lower
>>
>>
\layout { indent = 2.5\cm }
}

\score {
  <<
    \context Staff = "trumpet" \with { midiInstrument = "trumpet" }
    \trompette
    \context DrumStaff = "tambourin"
    \tambourin
    \context Staff = "piano"
    <<
      \upper
      \lower
    >>
  >>
  \midi { \tempo 4 = 72 }
}

```

The musical score consists of three staves. The top staff is for Trumpet Bb, the middle for Tambourine, and the bottom for Piano. The key signature has two flats (Bb major), and the time signature is 2/4. The tempo is marked 'Comodo' and 'p grazioso'. The Trumpet Bb part begins with a whole rest, followed by a series of eighth and sixteenth notes, including a triplet. The Tambourine part plays a steady eighth-note pattern. The Piano part plays a series of chords, starting with a piano (p) dynamic.

## Text headword

Text headword.

% L. v. Beethoven, Op. 110

% Piano sonata 31

% measures 1 - 7

```
\include "english.ly"
```

```

\new PianoStaff <<
  \new Staff = "right hand" {
    \clef treble
    \key af \major
    \time 3/4
    \tempo "Moderato cantabile molto espressivo"

    <c'' af'>4.( <af' ef'>8 ) q8.[ q16] |
    <df'' g'>4 <bf' g'>2 |
    <af' ef''>4.( <af' df''>8[] <af' ef''>-.(<af' f''>-.)] |
    << { ef''8.([ d''16]) df''8\trill\fermata ~
      \oneVoice df''32[ c'' df'' ef''] }\\
      { g'4 g'8 s } >>
    \grace { df''32[ ef''] } f''8[ ef''16 df''] |

%
% 5
%
    c''4.( ef''4 af''8) |
    af''4( g''2) |
    bf''4.( g''4 ef''8) |
  }

\new Dynamics {
  s2.-\tweak padding #-1
  -\tweak baseline-skip #0
  -\markup \center-column {
    \whiteout \line { \dynamic p \italic { con amabilità } }
    \line { \hspace #3 (sanft) } } |
  s2. |
  s2.\< |
  s8..\p s32\< s16..\> s64\! s8 s4\> |

%
% 5
%
  s2.*3\! |
}

\new Staff = "left hand" {
  \clef bass
  \key af \major
  \time 3/4

  <af, ef>4. \stemUp <c ef>8 q8.[ q16] \stemNeutral |
  <bf, ef>4 <df ef>2 |
  << { ef8[( af c' bf) c' df'] } \\
    { c4.( f8[] ef8-.(<df-.) ) } >> |
  <ef bf>4 q8_\fermata r r4 | \clef treble

%
% 5
%
  af16[ <c' ef'> q q] af[ <c' ef'> q q] af[ <c' ef'> q q] |
  bf16[ <df' ef'> q q] bf[ <df' ef'> q q] bf[ <df' ef'> q q] |

```

```

df'16[ <ef' g' bf'> q q] df'[ <ef' g' bf'> q q]
df'[ <ef' g' bf'> q q] |
}
>>

```

**Moderato cantabile molto espressivo**

## Unfretted headword

Unfretted headword.

```

% David Séverin
% Les cinq pieds (2007)
% for violon solo
% (this extract is in the public domain)

```

```

% Abbreviations
db = \markup { \musicglyph "scripts.udownbow" }
dub = \markup { \musicglyph "scripts.udownbow" " "
               \musicglyph "scripts.uupbow" }
dubetc = \markup { \musicglyph "scripts.udownbow" " "
                  \musicglyph "scripts.uupbow" "..."}

ub = \markup { \musicglyph "scripts.uupbow" }
udb = \markup { \musicglyph "scripts.uupbow" " "
               \musicglyph "scripts.udownbow" }
udbetc = \markup { \musicglyph "scripts.uupbow" " "
                  \musicglyph "scripts.udownbow" "..."}

accel = \markup \tiny \italic \bold "accel..."
ritar = \markup \tiny \italic \bold "ritar..."

% Strings
svib = \markup \small "s. vib."
pvib = \markup \small "p. vib."

```

```

mvib = \markup \small "m. vib."
sulp = \markup \small "s.p."
norm = \markup \small "n."

quatre = \markup \teeny "IV"

% Shifting Notes
shift = \once \override NoteColumn.force-hshift = 0.9
shifta = \once \override NoteColumn.force-hshift = 1.2

% Hairpin
aniente = \once \override Hairpin.circled-tip = ##t

% Tuplets
tupletbp = \once \override Staff.TupletBracket.padding = 2.25

% Functions
#(define-markup-command (colmark layout props args) (markup-list?)
  (let ((entries (cons (list '(baseline-skip . 2.3)) props)))
    (interpret-markup layout entries
      (make-column-markup (map (lambda (arg)
                                (markup arg))
                              (reverse args))))))

% Instruments
ViolinSolo = \relative c' {
  \set Score.rehearsalMarkFormatter = #format-mark-box-numbers
  \override Score.VoltaBracket.font-family = #'sans
  \override Score.VoltaBracket.extra-offset = #'(0 . 1)
  \override SpacingSpanner.uniform-stretching = ##t

  \voiceOne

  % Measure 1
  \time 25/8 \mark \default
  r2~\markup \colmark { \italic "fatigué" " " \bold "lentement"} r4 r r8
  <<
    { \shift d2\glissando~\markup \colmark { \quatre \dubetc \svib }
      \shifta e1 } \\\
    { d2\open\mf\< ~ \aniente d1\!\>
      r4 r\!\~\markup \colmark { " " \fermata } }
  >>

  % Measure 2
  \time 7/4
  \set Score.repeatCommands = #'((volta "1) n.          2) s.p."))
  <<
    { \shift d2\glissando~\markup \colmark { \quatre \udbetc }
      \shifta e1 } \\\
    { d2\open\mf\< ~ d1\!\> ~ d4\!\~\markup \colmark { " " \fermata } }
  >>
  \set Score.repeatCommands = #'((volta #f))

```

```

% Measure 3
\time 15/4
<<
  { \shift d2\glissando^\markup \colmark { \quatre \dubetc \pvib \norm }
    \shifta e1\glissando d2 } \
  { d2\open\mf\< ~ d1 ~ d2\ff ~ d1\> ~
    d2^\markup \colmark { " " " " \svib } ~ d4\pp }
>>
\break

% Measure 4
\time 4/4 \stemUp \tupletDown
\tuplet 3/2 { d4 ^\markup \colmark { \quatre \db \accel } d d }
\tuplet 3/2 { d4 ^\markup \colmark { " " \db \sulp } d d }

% Measure 5
\time 5/4
\tupletbp \tuplet 3/2 {
  d8\mf\<^\markup \colmark { \quatre \db \norm } d_\open d }
\tupletbp \tuplet 3/2 {
  d8^\markup \colmark { " " \db \sulp } d_\open d }
\tupletbp \tuplet 3/2 {
  d16^\markup \colmark { " " \db \norm } d_\open d d d_\open d }
d2\ff\>^\markup \colmark { " " \pvib }

% Measure 6
\time 5/8
\once \override Beam.grow-direction = #RIGHT % \featherDurations 2/3
d16-.[ d-. d-. d-. d-. d-. d-. d-. d-.]
\break

% Measure 7
\time 7/4
\tupletbp \tuplet 3/2 {
  d16^\markup \colmark { \quatre } d_\open d d d_\open d }
\tupletbp \tuplet 3/2 {
  d8^\markup \colmark { " " \db } d_\open d }
\tupletbp \tuplet 3/2 {
  d8^\markup \colmark { " " \db " " \sulp } d_\open d }
\tuplet 3/2 { d4^\markup \colmark { \quatre \db \ritar \norm } d d }
\tuplet 3/2 { d4^\markup \colmark { " " \db " " \sulp } d d\ppp ~ }

% Measure 8
d4^\markup \colmark { " " " " \pvib \norm } deh2 d dih \<

% Measure 9
<<
  { \shift d2\glissando^\markup \colmark { \quatre } \shifta e1 } \
  { d2\open ~ d1^\markup \colmark { " " " " \mvib } }
>>
\breathe r4\!
}

```

```

\score {
  <<
    \new Staff \relative c' \ViolinSolo
    \hide Score.Rest
    \set Score.measureBarType = ""
  >>

  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
    }
    \context {
      \Score
      \remove "Bar_number_engraver"
    }
  }
}

\paper {
  system-system-spacing.padding = 5
}

```

**1** **lentement**

*fatigué* s. vib. IV V ... 1) n. 2) s.p. n. p. vib. s. vib.

*mf* *mf* *mf* *ff* *pp*

*accel...* s.p. n. s.p. n. p. vib.

*mf* *ff*

s.p. n. s.p. n. p. vib. m. vib.

*ritar...*

*ppp*

## Vocal headword

Vocal headword.

% L. van Beethoven, op. 125

```

% Symphony No. 9 in D minor
% Finale measures 216 - 236
% Text: L. van Beethoven (introduction), F. von Schiller ("Ode to Joy")

\score {
  \new Staff \relative c' {
    \override Score.BarNumber.self-alignment-X = #LEFT
    \set Score.currentBarNumber = 216
    \set Score.barNumberVisibility = #all-bar-numbers-visible

    \autoBeamOff
    \clef bass \key d \minor \time 3/4
    \tempo "Recitativo"

    r4^\markup { \small Baritono } r a |
    \grace a8 e'2. ~ |
    e4( d8[ cis d e]) |
    e4 g, r8 g |
    bes2 a8 e |
    f4 f r |
    R2.*2 |

    gis2 gis4 |
    r4 d'4. b8 |
    b4 gis8\tweak height-limit #4 ([ a b cis] |
    e8[ d cis d]) b([ gis]) |
    e8 d d4 fis8([ e]) |
    d4 cis r \bar "||"

    \key d \major
    r4 r a' |
    d4.( e8[ fis e]) |
    e([ d]) d([ cis d a]) |
    g8([ fis]) fis([ e d c]) |
    c8([ b]) g'2~ |
    % put fermata closer to staff
    \once \override Script.outside-staff-priority = #1
    g4.\fermata ^\markup { \small \italic "ad libitum" } e8[ cis!] d |
    d8 a a4 r \bar "||"
  }

  \addlyrics {
    O Freun -- _ de, nicht die -- _ se Tö -- ne!
    Son -- dern laßt uns an -- _ ge -- neh -- me -- re an -- stim -- men,
    und freu -- _ _ _ _ _ den -- vol -- le -- re!
  }
}

```



```
\score {
  \new StaffGroup <<
    \new Staff \with { instrumentName = "Flauto I,II" }
    \relative c' {
      \tweak padding 3.5 \tweak font-size 1 \tempo "Moderato assai"
      \key d \major
      \time 2/4
      \compressMMRests R2*2
      <d a>16-. \p <cis g>- . <d a>- . <cis g>- . <d a>8-. <cis g>- .
      <e a,>- . \< <d a>32( <fis d> <a fis> <d a> <fis d>4--)\mf
      <g d>16-. <fis cis>- . <g d>- . <fis cis>- .
      <e b>(\> <d a>) <a fis>- . <fis d>- . \!
      <d bes>4--\sf \acciaccatura {<d' bes>8} <cis a>4--\mf
    }
  \new Staff \with { instrumentName = "Flauto III" }
  \relative c' {
    \key d \major
    \time 2/4
    \compressMMRests R2*2_\markup{Gr.Fl.}
    fis16-. \p e-. fis-. e-. fis8-. e-.
    g8-. \< fis32( a d fis a4--)\mf
    b16-. a-. b-. a-. g(\> fis) d-. a-. \!
    g4--\sf \acciaccatura fis'8 g4--\mf
  }
}>>
```

```
\layout {
  indent = 2\cm
}
```

**Moderato assai**

Flauto I,II

Flauto III

Gr.Fl.

*p* *mf* *sf* *mf*

*p* *mf* *sf* *mf*

## 25 MIDI

See also Section “Creating MIDI output” in *Notation Reference*.

### Changing MIDI output to one channel per voice

When outputting MIDI, the default behavior is for each staff to represent one MIDI channel, with all the voices on a staff amalgamated. This minimizes the risk of running out of MIDI channels, since there are only 16 available per track.

However, by moving the `Staff_performer` to the `Voice` context, each voice on a staff can have its own MIDI channel, as is demonstrated by the following example: despite being on the same staff, two MIDI channels are created, each with a different `midiInstrument`.

```
\score {
  \new Staff <<
    \new Voice \relative c''' {
      \set midiInstrument = "flute"
      \voiceOne
      \key g \major
      \time 2/2
      r2 g-"Flute" ~
      g fis ~
      fis4 g8 fis e2 ~
      e4 d8 cis d2
    }
    \new Voice \relative c'' {
      \set midiInstrument = "clarinet"
      \voiceTwo
      b1-"Clarinet"
      a2. b8 a
      g2. fis8 e
      fis2 r
    }
  >>
  \layout { }
  \midi {
    \context {
      \Staff
      \remove "Staff_performer"
    }
    \context {
      \Voice
      \consists "Staff_performer"
    }
  }
  \tempo 2 = 72
}
```



## 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 { }
}
```



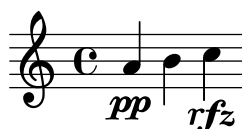
## Creating custom dynamics in MIDI output

The following example shows how to create a dynamic marking, not included in the default list, and assign a specific value to it so that it affects MIDI output.

The dynamic mark `\rfz` gets value 0.9.

```
#(define (myDynamics dynamic)
  (if (equal? dynamic "rfz")
      0.9
      (default-dynamic-absolute-volume dynamic)))

\score {
  \new Staff {
    \set Staff.midiInstrument = "cello"
    \set Score.dynamicAbsoluteVolumeFunction = #myDynamics
    \new Voice {
      \relative {
        a'4\pp b c-\rfz
      }
    }
  }
  \layout {}
  \midi {}
}
```



## Demo of MIDI instruments

Problem: How to know which `midiInstrument` values would be best for your composition?

Solution: A LilyPond demo file. You have to compile this snippet by yourself and listen to the created MIDI output file.

```
melody = \relative c' {
  \tempo 4 = 150
  c4.\mf g c16 b' c d
  e16 d e f g4 g'4 r
  R1
}

\score {
  \new Voice \melody
  \layout { }
}

\score {
  \new Voice {
    r\mf
    % 1-8 keyboard
    \set Staff.midiInstrument = "acoustic grand" \melody
    \set Staff.midiInstrument = "bright acoustic" \melody
    \set Staff.midiInstrument = "electric grand" \melody
    \set Staff.midiInstrument = "honky-tonk" \melody
    \set Staff.midiInstrument = "electric piano 1" \melody
    \set Staff.midiInstrument = "electric piano 2" \melody
    \set Staff.midiInstrument = "harpsichord" \melody
    \set Staff.midiInstrument = "clav" \melody

    % 9-16 chrom percussion
    \set Staff.midiInstrument = "celesta" \melody
    \set Staff.midiInstrument = "glockenspiel" \melody
    \set Staff.midiInstrument = "music box" \melody
    \set Staff.midiInstrument = "vibraphone" \melody
    \set Staff.midiInstrument = "marimba" \melody
    \set Staff.midiInstrument = "xylophone" \melody
    \set Staff.midiInstrument = "tubular bells" \melody
    \set Staff.midiInstrument = "dulcimer" \melody

    % 17-24 organ
    \set Staff.midiInstrument = "drawbar organ" \melody
    \set Staff.midiInstrument = "percussive organ" \melody
    \set Staff.midiInstrument = "rock organ" \melody
    \set Staff.midiInstrument = "church organ" \melody
    \set Staff.midiInstrument = "reed organ" \melody
    \set Staff.midiInstrument = "accordion" \melody
    \set Staff.midiInstrument = "harmonica" \melody
    \set Staff.midiInstrument = "concertina" \melody

    % 25-32 guitar
    \set Staff.midiInstrument = "acoustic guitar (nylon)" \melody
```

```

\set Staff.midiInstrument = "acoustic guitar (steel)" \melody
\set Staff.midiInstrument = "electric guitar (jazz)" \melody
\set Staff.midiInstrument = "electric guitar (clean)" \melody
\set Staff.midiInstrument = "electric guitar (muted)" \melody
\set Staff.midiInstrument = "overdriven guitar" \melody
\set Staff.midiInstrument = "distorted guitar" \melody
\set Staff.midiInstrument = "guitar harmonics" \melody

```

*% 33-40 bass*

```

\set Staff.midiInstrument = "acoustic bass" \melody
\set Staff.midiInstrument = "electric bass (finger)" \melody
\set Staff.midiInstrument = "electric bass (pick)" \melody
\set Staff.midiInstrument = "fretless bass" \melody
\set Staff.midiInstrument = "slap bass 1" \melody
\set Staff.midiInstrument = "slap bass 2" \melody
\set Staff.midiInstrument = "synth bass 1" \melody
\set Staff.midiInstrument = "synth bass 2" \melody

```

*% 41-48 strings*

```

\set Staff.midiInstrument = "violin" \melody
\set Staff.midiInstrument = "viola" \melody
\set Staff.midiInstrument = "cello" \melody
\set Staff.midiInstrument = "contrabass" \melody
\set Staff.midiInstrument = "tremolo strings" \melody
\set Staff.midiInstrument = "pizzicato strings" \melody
\set Staff.midiInstrument = "orchestral harp" \melody
\set Staff.midiInstrument = "timpani" \melody

```

*% 49-56 ensemble*

```

\set Staff.midiInstrument = "string ensemble 1" \melody
\set Staff.midiInstrument = "string ensemble 2" \melody
\set Staff.midiInstrument = "synthstrings 1" \melody
\set Staff.midiInstrument = "synthstrings 2" \melody
\set Staff.midiInstrument = "choir aahs" \melody
\set Staff.midiInstrument = "voice oohs" \melody
\set Staff.midiInstrument = "synth voice" \melody
\set Staff.midiInstrument = "orchestra hit" \melody

```

*% 57-64 brass*

```

\set Staff.midiInstrument = "trumpet" \melody
\set Staff.midiInstrument = "trombone" \melody
\set Staff.midiInstrument = "tuba" \melody
\set Staff.midiInstrument = "muted trumpet" \melody
\set Staff.midiInstrument = "french horn" \melody
\set Staff.midiInstrument = "brass section" \melody
\set Staff.midiInstrument = "synthbrass 1" \melody
\set Staff.midiInstrument = "synthbrass 2" \melody

```

*% 65-72 reed*

```

\set Staff.midiInstrument = "soprano sax" \melody
\set Staff.midiInstrument = "alto sax" \melody
\set Staff.midiInstrument = "tenor sax" \melody

```

```

\set Staff.midiInstrument = "baritone sax" \melody
\set Staff.midiInstrument = "oboe" \melody
\set Staff.midiInstrument = "english horn" \melody
\set Staff.midiInstrument = "bassoon" \melody
\set Staff.midiInstrument = "clarinet" \melody

% 73-80 pipe
\set Staff.midiInstrument = "piccolo" \melody
\set Staff.midiInstrument = "flute" \melody
\set Staff.midiInstrument = "recorder" \melody
\set Staff.midiInstrument = "pan flute" \melody
\set Staff.midiInstrument = "blown bottle" \melody
\set Staff.midiInstrument = "shakuhachi" \melody
\set Staff.midiInstrument = "whistle" \melody
\set Staff.midiInstrument = "ocarina" \melody

% 81-88 synth lead
\set Staff.midiInstrument = "lead 1 (square)" \melody
\set Staff.midiInstrument = "lead 2 (sawtooth)" \melody
\set Staff.midiInstrument = "lead 3 (calliope)" \melody
\set Staff.midiInstrument = "lead 4 (chiff)" \melody
\set Staff.midiInstrument = "lead 5 (charang)" \melody
\set Staff.midiInstrument = "lead 6 (voice)" \melody
\set Staff.midiInstrument = "lead 7 (fifths)" \melody
\set Staff.midiInstrument = "lead 8 (bass+lead)" \melody

% 89-96 synth pad
\set Staff.midiInstrument = "pad 1 (new age)" \melody
\set Staff.midiInstrument = "pad 2 (warm)" \melody
\set Staff.midiInstrument = "pad 3 (polysynth)" \melody
\set Staff.midiInstrument = "pad 4 (choir)" \melody
\set Staff.midiInstrument = "pad 5 (bowed)" \melody
\set Staff.midiInstrument = "pad 6 (metallic)" \melody
\set Staff.midiInstrument = "pad 7 (halo)" \melody
\set Staff.midiInstrument = "pad 8 (sweep)" \melody

% 97-104 synth effects
\set Staff.midiInstrument = "fx 1 (rain)" \melody
\set Staff.midiInstrument = "fx 2 (soundtrack)" \melody
\set Staff.midiInstrument = "fx 3 (crystal)" \melody
\set Staff.midiInstrument = "fx 4 (atmosphere)" \melody
\set Staff.midiInstrument = "fx 5 (brightness)" \melody
\set Staff.midiInstrument = "fx 6 (goblins)" \melody
\set Staff.midiInstrument = "fx 7 (echoes)" \melody
\set Staff.midiInstrument = "fx 8 (sci-fi)" \melody

% 105-112 ethnic
\set Staff.midiInstrument = "sitar" \melody
\set Staff.midiInstrument = "banjo" \melody
\set Staff.midiInstrument = "shamisen" \melody
\set Staff.midiInstrument = "koto" \melody
\set Staff.midiInstrument = "kalimba" \melody

```

```

\set Staff.midiInstrument = "bagpipe" \melody
\set Staff.midiInstrument = "fiddle" \melody
\set Staff.midiInstrument = "shanai" \melody

% 113-120 percussive
\set Staff.midiInstrument = "tinkle bell" \melody
\set Staff.midiInstrument = "agogo" \melody
\set Staff.midiInstrument = "steel drums" \melody
\set Staff.midiInstrument = "woodblock" \melody
\set Staff.midiInstrument = "taiko drum" \melody
\set Staff.midiInstrument = "melodic tom" \melody
\set Staff.midiInstrument = "synth drum" \melody
\set Staff.midiInstrument = "reverse cymbal" \melody

% 121-128 sound effects
\set Staff.midiInstrument = "guitar fret noise" \melody
\set Staff.midiInstrument = "breath noise" \melody
\set Staff.midiInstrument = "seashore" \melody
\set Staff.midiInstrument = "bird tweet" \melody
\set Staff.midiInstrument = "telephone ring" \melody
\set Staff.midiInstrument = "helicopter" \melody
\set Staff.midiInstrument = "applause" \melody
\set Staff.midiInstrument = "gunshot" \melody
}
\midi { }
}

```



## Replacing default MIDI instrument equalization

The default MIDI instrument equalizer can be replaced by setting the `instrumentEqualizer` property in the Score context to a user-defined Scheme procedure that uses a MIDI instrument name as its argument along with a pair of fractions indicating the minimum and maximum volumes, respectively, to be applied to that specific instrument.

The following example sets the minimum and maximum volumes for flute and clarinet.

```

#(define my-instrument-equalizer-alist '())

#(set! my-instrument-equalizer-alist
  (append
    '(("flute" . (0.7 . 0.9))
      ("clarinet" . (0.3 . 0.6)))
    my-instrument-equalizer-alist))

#(define (my-instrument-equalizer s)
  (let ((entry (assoc s my-instrument-equalizer-alist)))
    (if entry
      (cdr entry))))

```



```

\score {
  <<
    \new Staff {
      \key g \major
      \time 2/2
      \set Score.instrumentEqualizer = #my-instrument-equalizer
      \set Staff.midiInstrument = "flute"
      \new Voice \relative {
        r2 g''\mp g fis~
        4 g8 fis e2~
        4 d8 cis d2
      }
    }
    \new Staff {
      \key g \major
      \set Staff.midiInstrument = "clarinet"
      \new Voice \relative {
        b'1\p a2. b8 a
        g2. fis8 e
        fis2 r
      }
    }
  >>
  \layout { }
  \midi { }
}

```



## 26 Non-music

### Aligning and centering instrument names

The horizontal alignment of instrument names is tweaked by changing the `self-alignment-X` property of the `InstrumentName` grob (usually in the `Staff` context). The `\layout` variables `indent` and `short-indent` define the space in which the instrument names are aligned before the first and the following systems, respectively.

```
\paper {
  left-margin = 3\cm
}

\new StaffGroup <<
  \new Staff \with {
    \override InstrumentName.self-alignment-X = #LEFT
    instrumentName = \markup \left-column { "Left aligned"
                                           "instrument name" }

    shortInstrumentName = "Left"
  } {
    c''1 \break c''1
  }

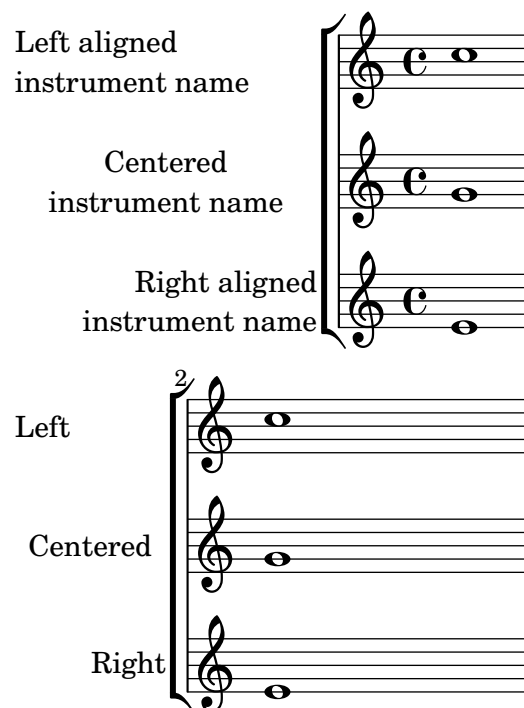
  \new Staff \with {
    \override InstrumentName.self-alignment-X = #CENTER
    instrumentName = \markup \center-column { Centered
                                           "instrument name" }

    shortInstrumentName = "Centered"
  } {
    g'1 g'1
  }

  \new Staff \with {
    \override InstrumentName.self-alignment-X = #RIGHT
    instrumentName = \markup \right-column { "Right aligned"
                                           "instrument name" }

    shortInstrumentName = "Right"
  } {
    e'1 e'1
  }
}
>>

\layout {
  indent = 4\cm
  short-indent = 2\cm
  line-width = 6.5\cm
}
```



## Demonstrating all \header fields

A demonstration of all header fields that LilyPond defines by default. Thanks to setting `print-all-headers` to `#t`, much more fields as usual are displayed, indicating the hierarchy of `\header` blocks.

```
\paper {
  #(set-paper-size "a6" 'landscape)
  print-all-headers = ##t
}

\book {
  \header {
    title = "title"
    subtitle = "subtitle"
    composer = "composer"
    arranger = "arranger"
    instrument = "instrument"
    meter = "meter"
    opus = "opus"
    piece = "piece"
    poet = "poet"
    copyright = "copyright"
    tagline = "tagline"
  }

  \bookpart {
    \score {
      \relative c'' { c1 | c | c | c }


      \header {
        title = "localtitle"
      }
    }
  }
}
```

```

    subtitle = "localsubtitle"
    composer = "localcomposer"
    arranger = "localarranger"
    instrument = "localinstrument"
    meter = "localmeter"
    opus = "localopus"
    piece = "localpiece"
    poet = "localpoet"
    copyright = "localcopyright"
    tagline = "localtagline"
  }
}
}
}

```

	<b>title</b>	
	<b>subtitle</b>	
poet	<b>instrument</b>	composer
meter		arranger
	<b>localtitle</b>	
	<b>localsubtitle</b>	
localpoet	<b>localinstrument</b>	localcomposer
localmeter		localarranger
localpiece		localopus

	copyright
	tagline

## 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 gets written to `stderr` but is not shown in the music. If output to `stdout` is wanted instead, omit the code `(current-error-port)` from the commands.

```

#(print-keys-verbose 'piccolo (current-error-port))
#(print-keys-verbose 'flute (current-error-port))
#(print-keys-verbose 'flute-b-extension (current-error-port))
#(print-keys-verbose 'tin-whistle (current-error-port))
#(print-keys-verbose 'oboe (current-error-port))
#(print-keys-verbose 'clarinet (current-error-port))
#(print-keys-verbose 'bass-clarinet (current-error-port))
#(print-keys-verbose 'low-bass-clarinet (current-error-port))
#(print-keys-verbose 'saxophone (current-error-port))
#(print-keys-verbose 'soprano-saxophone (current-error-port))
#(print-keys-verbose 'alto-saxophone (current-error-port))

```

```
#(print-keys-verbose 'tenor-saxophone (current-error-port))  
#(print-keys-verbose 'baritone-saxophone (current-error-port))  
#(print-keys-verbose 'bassoon (current-error-port))  
#(print-keys-verbose 'contrabassoon (current-error-port))
```

```
\score {c''1}
```



## 27 Paper and layout

See also Section “Spacing issues” in *Notation Reference*.

### Aligning and centering instrument names

The horizontal alignment of instrument names is tweaked by changing the `self-alignment-X` property of the `InstrumentName` grob (usually in the `Staff` context). The `\layout` variables `indent` and `short-indent` define the space in which the instrument names are aligned before the first and the following systems, respectively.

```
\paper {
  left-margin = 3\cm
}

\new StaffGroup <<
  \new Staff \with {
    \override InstrumentName.self-alignment-X = #LEFT
    instrumentName = \markup \left-column { "Left aligned"
                                           "instrument name" }

    shortInstrumentName = "Left"
  } {
    c''1 \break c''1
  }

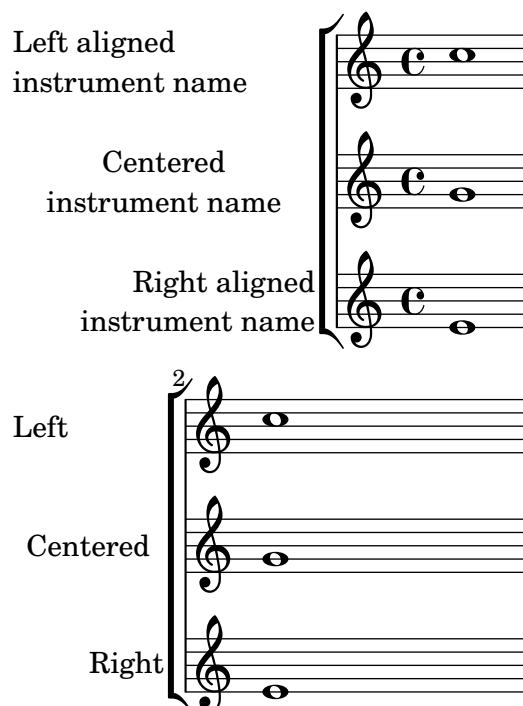
  \new Staff \with {
    \override InstrumentName.self-alignment-X = #CENTER
    instrumentName = \markup \center-column { Centered
                                           "instrument name" }

    shortInstrumentName = "Centered"
  } {
    g'1 g'1
  }

  \new Staff \with {
    \override InstrumentName.self-alignment-X = #RIGHT
    instrumentName = \markup \right-column { "Right aligned"
                                           "instrument name" }

    shortInstrumentName = "Right"
  } {
    e'1 e'1
  }
}
>>

\layout {
  indent = 4\cm
  short-indent = 2\cm
  line-width = 6.5\cm
}
```



## Arranging separate lyrics on a single line

Sometimes you may want to put lyrics for different performers on a single line: where there is rapidly alternating text, for example. This snippet shows how it can be done with adjusting the `nonstaff-nonstaff-spacing` property of the `VerticalAxisGroup` grob.

```
\layout {
  \context {
    \Lyrics
    \override VerticalAxisGroup
      .nonstaff-nonstaff-spacing
      .minimum-distance = ##f
  }
}

aliceSings = \markup { \smallCaps "Alice" }
eveSings = \markup { \smallCaps "Eve" }

<<
\new Staff <<
  \new Voice = "alice" {
    f'4^\aliceSings g' r2 |
    s1 |
    f'4^\aliceSings g' r2 |
    s1 | \break
    % ...

    \voiceOne
    s2 a'8^\aliceSings a' b'4 |
    \oneVoice
    g'1
  }
}
```

```

\new Voice = "eve" {
  s1 |
  a'2^\eveSings g' |
  s1 |
  a'2^\eveSings g'
  % ...

  \voiceTwo
  f'4^\eveSings a'8 g' f'4 e' |
  \oneVoice
  s1
}
>>

\new Lyrics \lyricsto "alice" {
  may -- be
  sec -- ond
  % ...
  Shut up, you fool!
}

\new Lyrics \lyricsto "eve" {
  that the
  words are
  % ...
  ...and then I was like--
}
>>

```

The musical score is written on four staves. The first staff contains the lyrics 'may - be' (Alice) and 'that the' (Eve). The second staff contains 'sec - ond' (Alice) and 'words are' (Eve). The third staff contains '...and then I' (Eve) and 'Shut up, you like--' (Alice). The fourth staff contains 'was' (Alice) and 'fool!' (Eve). The lyrics are written below the staves, and the names 'ALICE' and 'EVE' are written above the staves to indicate which character is singing.

## Book parts

`\bookpart` can be used to split a book into several parts. Each part last page can be affected by `ragged-last-bottom`. Header and footer markups can detect a part's last page to differentiate with the book's last page.

```
#(set-default-paper-size "a6")
```

```

\book {
  %% book paper, which is inherited by all children bookparts
  \paper {
    ragged-last-bottom = ##t
    %% Page footer: add a different part-tagline at part last page
  }
}

```



```

oddFooterMarkup = \markup {
  \column {
    \fill-line {
      %% Copyright header field only on book first page.
      \if \on-first-page \fromproperty #'header:copyright
    }
    \fill-line {
      %% Part tagline header field only on each part last page.
      \if \on-last-page-of-part \fromproperty #'header:parttagline
    }
    \fill-line {
      %% Tagline header field only on book last page.
      \if \on-last-page \fromproperty #'header:tagline
    }
  }
}

%% book header, which is inherited by the first bookpart
\header {
  title = "Book title"
  copyright = "Copyright line on book first page"
  parttagline = "Part tagline"
  tagline = "Book tagline"
}

\bookpart {
  %% a different page breaking function may be used on each part
  \paper { page-breaking = #ly:minimal-breaking }
  \header { subtitle = "First part" }
  \markup { The first book part }
  \markup { a page break }
  \pageBreak
  \markup { first part last page }
  \markup \wordwrap {
    with ragged-last-bottom (see the space below this text) }
}

\bookpart {
  \header { subtitle = "Second part" }
  { c'4 }
}

```

# **Book title**

## **First part**

The first book part

a page break

Copyright line on book first page

2

first part last page  
with ragged-last-bottom (see the space below this  
text)

Part tagline

3

**Book title****Second part**

Part tagline

Book tagline

**Changing the staff size**

Though the simplest way to resize staves is to use  `#(set-global-staff-size size)`, an individual staff's size can be changed by scaling the properties `staff-space` and `fontSize`.

```
<<
  \new Staff {
    \relative c'' {
      \dynamicDown
      c8\ff c c c c c c c
    }
  }
  \new Staff \with {
    fontSize = #-3
    \override StaffSymbol.staff-space = #(magstep -3)
  } {
    \clef bass
    c8 c c c c\f c c c
  }
>>
```



## Clip systems

This code shows how to clip (extract) snippets from a full score.

This file needs to be run separately with `-dclip-systems`; the snippets page may not adequately show the results. The result will be files named `'base-from-start-to-end[-count].eps'`.

If system starts and ends are included, they include extents of the System grob, e.g., instrument names.

Grace notes at the end point of the region are not included.

Regions can span multiple systems. In this case, multiple EPS files are generated.

```
#(set-default-paper-size "a6" 'landscape)
\layout {
  indent = 2.4\cm
}

#(ly:set-option 'clip-systems)
#(ly:set-option 'separate-page-formats "ps")
#(define output-suffix "1")

origScore = \new Staff \with { instrumentName = "Instrument" }
\relative c' {
  c1
  d1
  \grace c16 e1
  \key d \major
  f1 \break
  \clef bass
  g,1
  fis1
}

\book {
  \score {
    \origScore
    \layout {
      % Each clip-region is a (START . END) pair
      % where both are rhythmic locations. Syntax:
      %
      % (make-rhythmic-locations BAR-NUMBER NUM DEN)
      %
      % means NUM/DEN whole-notes into bar numbered BAR-NUMBER

      clip-regions = #(list (cons (make-rhythmic-location 2 0 1)
                                (make-rhythmic-location 4 0 1)))
    }
  }
}
```

```

      (cons (make-rhythmic-location 0 0 1)
            (make-rhythmic-location 4 0 1))
      (cons (make-rhythmic-location 0 0 1)
            (make-rhythmic-location 6 0 1)))
    }
  }
}

#(ly:set-option 'clip-systems #f)
#(ly:set-option 'separate-page-formats #f)
#(define output-suffix #f)

\book {
  \score { \origScore }
  \markup { \bold \fontsize #6 clips }
  \score {
    \lyrics {
      "from-2.0.1-to-4.0.1-clip.eps"
      \markup \epsfile #X #30.0
        #(format #f "~a-1-from-2.0.1-to-4.0.1-clip.eps"
          (ly:parser-output-name))
    }
  }
}

```



LilyPond v2.25.32

## Creating blank staves

To create blank staves, generate empty measures then remove the `Bar_number_engraver` from the `Score` context, and the `Time_signature_engraver`, `Clef_engraver` and `Bar_engraver` from the `Staff` context.

```

#(set-global-staff-size 10) % for the documentation
% #(set-global-staff-size 20) % for letter and A4

```

```

\book {
  \score {
    { \repeat unfold 12 { s1 \break } }

    \layout {
      indent = 0
      \context {
        \Staff
        \remove "Time_signature_engraver"
        \remove "Clef_engraver"
        \remove "Bar_engraver"
      }
      \context {
        \Score
        \remove "Bar_number_engraver"
      }
    }
  }

  % for the documentation
  \paper {
    #(set-paper-size "a6")
    ragged-last-bottom = ##f
    line-width = 90\mm
    left-margin = 7.5\mm
    bottom-margin = 5\mm
    top-margin = 5\mm
    tagline = ##f
  }

  % uncomment these lines for "letter" size
  %{
  \paper {
    #(set-paper-size "letter")
    ragged-last-bottom = ##f
    line-width = 7.5\in
    left-margin = 0.5\in
    bottom-margin = 0.25\in
    top-margin = 0.25\in
    tagline = ##f
  }
  %}

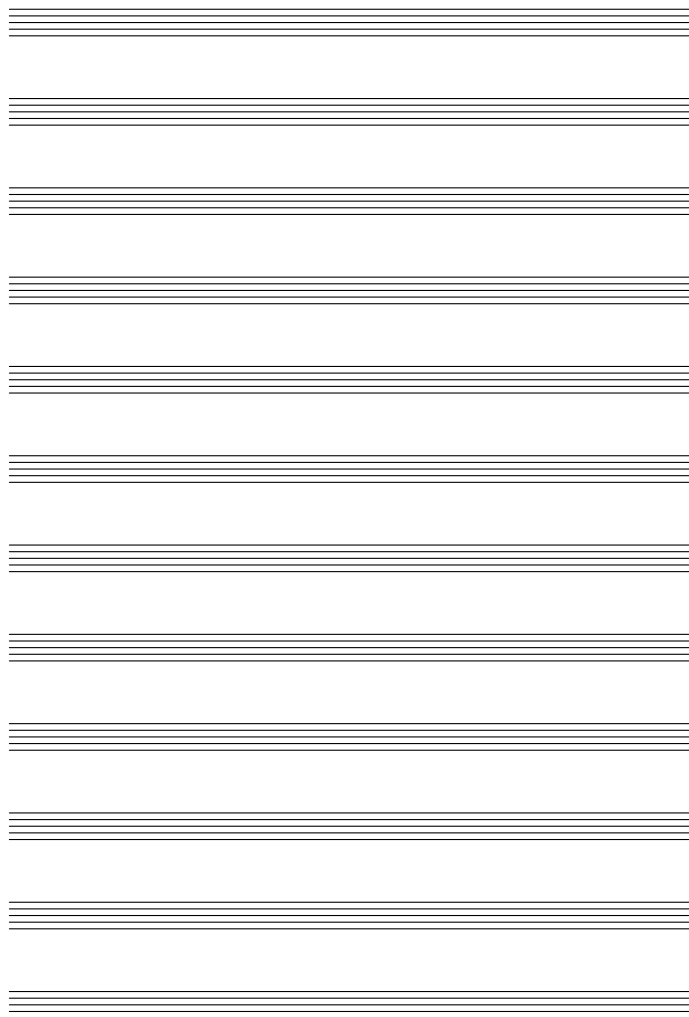
  % uncomment these lines for "A4" size
  %{
  \paper {
    #(set-paper-size "a4")
    ragged-last-bottom = ##f
    line-width = 180\mm
    left-margin = 15\mm
    bottom-margin = 10\mm
  }
  %}

```

```

    top-margin = 10\mm
    tagline = ##f
  }
  %}
}

```



## Demonstrating all \header fields

A demonstration of all header fields that LilyPond defines by default. Thanks to setting `print-all-headers` to `##t`, much more fields as usual are displayed, indicating the hierarchy of `\header` blocks.

```

\paper {
  #(set-paper-size "a6" 'landscape)
  print-all-headers = ##t
}

```

```

\book {
  \header {
    title = "title"
    subtitle = "subtitle"
    composer = "composer"
    arranger = "arranger"
  }
}

```



```

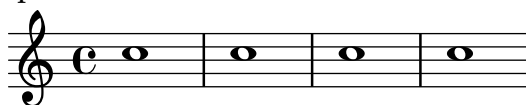
    instrument = "instrument"
    meter = "meter"
    opus = "opus"
    piece = "piece"
    poet = "poet"
    copyright = "copyright"
    tagline = "tagline"
}

\bookpart {
  \score {
    \relative c'' { c1 | c | c | c }

    \header {
      title = "localtitle"
      subtitle = "localsubtitle"
      composer = "localcomposer"
      arranger = "localarranger"
      instrument = "localinstrument"
      meter = "localmeter"
      opus = "localopus"
      piece = "localpiece"
      poet = "localpoet"
      copyright = "localcopyright"
      tagline = "localtagline"
    }
  }
}

```

	<b>title</b>	
	<b>subtitle</b>	
poet	<b>instrument</b>	composer
meter		arranger
	<b>localtitle</b>	
	<b>localsubtitle</b>	
localpoet	<b>localinstrument</b>	localcomposer
localmeter		localarranger
localpiece		localopus



copyright  
tagline

## Displaying a whole GrandStaff system if only one of its staves is alive

In many orchestral scores it is custom to not show staves for instruments that are silent for a while; this is called a ‘Frenched’ score. LilyPond provides this functionality via the `\RemoveEmptyStaves` command.

When they play again it is often preferred to show the staves of *all instruments of such a group*. This can be done by adding the `Keep_alive_together_engraver` to the grouping context (e.g., `GrandStaff` or `StaffGroup`).

In the example below the violins are silent in the second system. Only the first violin plays the last measure in the third system but the staff of the second violin is also displayed.

```
\score {
  <<
    \new Staff = "Staff_flute" \with {
      instrumentName = "Flute"
      shortInstrumentName = "Fl"
    } \relative c' {
      \repeat unfold 3 { c'4 c c c | c c c c | c c c c | \break }
    }

    \new StaffGroup = "StaffGroup_Strings" <<
      \new GrandStaff = "GrandStaff_violins" <<
        \new Staff = "StaffViolinI" \with {
          instrumentName = "Violin I"
          shortInstrumentName = "Vi I"
        } \relative c'' {
          a1 | R1*7 | \repeat unfold 12 a16 a4 |
        }
        \new Staff = "StaffViolinII" \with {
          instrumentName = "Violin II"
          shortInstrumentName = "Vi II"
        } \relative c' {
          e1 | R1*8 |
        }
      >>
    >>

    \new Staff = "Staff_cello" \with {
      instrumentName = "Cello"
      shortInstrumentName = "Ce"
    } \relative c {
      \clef bass \repeat unfold 9 { c1 } |
    }
  >>
}

\layout {
  indent = 3.0\cm
  short-indent = 1.5\cm

  \context {
    \GrandStaff
    \consists Keep_alive_together_engraver
```

```

}
\context {
  \Staff
  \RemoveEmptyStaves
}
}

```

The image displays three musical systems, each with a brace on the left side. The first system includes Flute, Violin I, Violin II, and Cello. The second system includes Flute and Cello. The third system includes Flute, Violin I, Violin II, and Cello. The systems are separated by a slash symbol, indicating the use of a system separator.

## Setting system separators

System separators can be inserted between systems. Any markup can be used, but `\slashSeparator` has been provided as a sensible default.

```
#(set-default-paper-size "a5")
```

```

\paper {
  system-separator-markup = \slashSeparator
  tagline = ##f
}

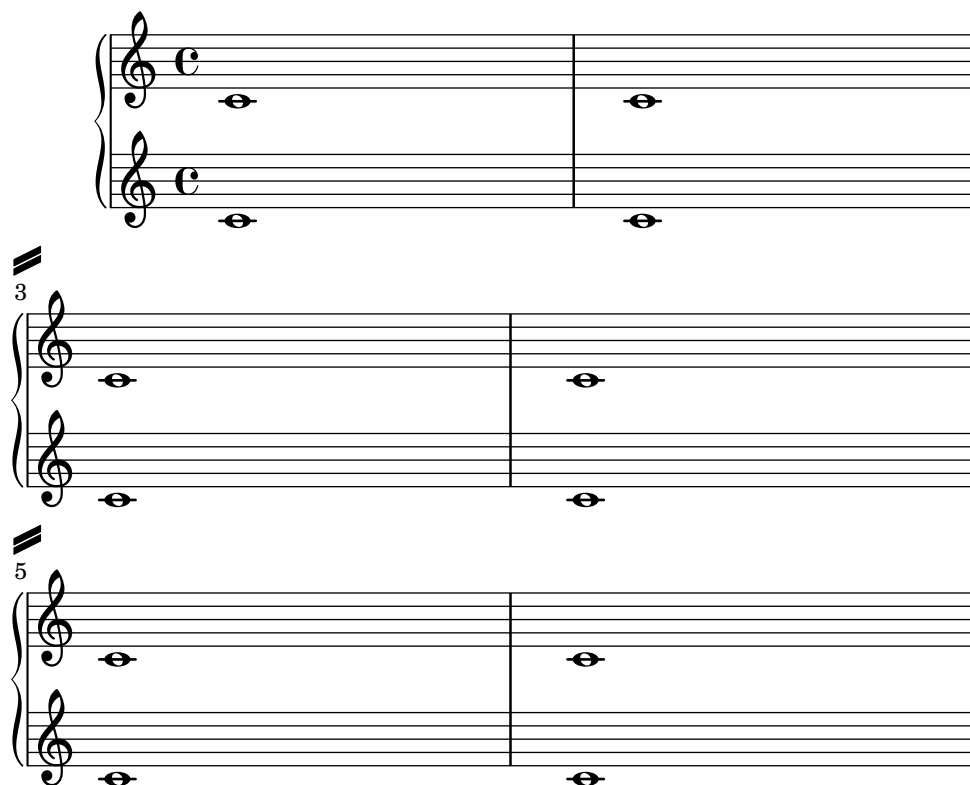
```

```

notes = \relative c' {
  c1 | c \break
  c1 | c \break
  c1 | c
}

\book {
  \score {
    \new GrandStaff <<
      \new Staff \notes
      \new Staff \notes
    >>
  }
}

```



## Table of contents

A table of contents is included using `\markuplist \table-of-contents`; its items are added with the `\tocItem` command.

```

#(set-default-paper-size "a7" 'landscape)
#(set-global-staff-size 11)

```

```

\paper {
  print-all-headers = ##t
}

\book {
  \markuplist \table-of-contents
}

```

```

\pageBreak

\tocItem \markup { The first score }
\score {
  {
    c'1 \pageBreak
    \mark \default \tocItem \markup { Mark A }
    d'1
  }
  \header { title = "First score" }
}
\pageBreak

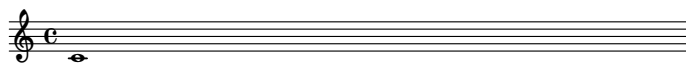
\tocItem \markup { The second score }
\score {
  { e'1 }
  \header { title = "Second score" }
}
}

```

## Table of Contents

The first score	2
Mark A	3
The second score	4

2

**First score**

3



4

**Second score**

## Vertically aligned StaffGroups without connecting SystemStartBar

This snippet shows how to achieve vertically aligned StaffGroups with a SystemStartBar for each StaffGroup, but without connecting them.

Note that this only works properly for music that can be printed as a single system.

```

#(set-global-staff-size 15)

\paper {
  ragged-right = ##f
  print-all-headers = ##t
  tagline = ##f
}

\layout {
  indent = 0

  \context {
    \StaffGroup
    \consists Text_mark_engraver
    \consists Staff_collecting_engraver
    systemStartDelimiterHierarchy =
      #'(SystemStartBrace (SystemStartBracket a b))
  }

  \context {
    \Score
    \remove Text_mark_engraver
    \remove Staff_collecting_engraver
    \override SystemStartBrace.style = #'bar-line
    \omit SystemStartBar
    \override SystemStartBrace.padding = #-0.1
    \override SystemStartBrace.thickness = #1.6
    \override StaffGrouper.staffgroup-staff-spacing.basic-distance = #15
  }
}

%%% EXAMPLE

txt =
\lyricmode {
  Wer4 nur den lie -- ben Gott läßt wal2 -- ten4
  und4 hof -- fet auf ihn al -- le Zeit2.
}

% First StaffGroup "exercise"

eI = \relative c' {
  \textMark \markup {
    \bold Teacher:
    This is a simple setting of the choral. Please improve it. }
  \key a \minor
  \time 4/4

```

```

\voiceOne

\partial 4 e4
a b c b
a b gis2
e4\fermata g! g f
e a a gis
a2.\fermata
\bar " : | ."
}

eII = \relative c' {
  \key a \minor
  \time 4/4
  \voiceTwo
  \partial 4 c4
  e e e gis
  a f e2
  b4 b d d
  c c d d
  c2.
  \bar " : | ."
}

eIII = \relative c' {
  \key a \minor
  \time 4/4
  \clef bass
  \voiceOne

  \partial 4 a4
  c b a b
  c d b2
  gis4 g g b
  c a f e
  e2.
}

eIV = \relative c' {
  \key a \minor
  \time 4/4
  \clef bass
  \voiceTwo

  \partial 4 a,4
  a' gis a e
  a, d e2
  e,4\fermata e' b g
  c f d e
  a,2.\fermata
  \bar " : | ."
}

```

```

exercise = \new StaffGroup = "exercise" <<
  \new Staff <<
    \new Voice \eI
    \new Voice \eII
  >>

  \new Lyrics \txt

  \new Staff <<
    \new Voice \eIII
    \new Voice \eIV
  >>
>>

% Second StaffGroup "simple Bach"

sbI = \relative c' {
  \textMark \markup { \bold" Pupil:" Here's my version! }
  \key a \minor
  \time 4/4
  \voiceOne

  \partial 4 e4
  a b c b
  a b gis2
  e4\fermata g! g f
  e a a gis
  a2.\fermata
  \bar ":|."
}

sbII = \relative c' {
  \key a \minor
  \time 4/4
  \voiceTwo
  \partial 4 c8 d
  e4 e e8 f g4
  f f e2
  b4 b8 c d4 d
  e8 d c4 b8 c d4
  c2.
  \bar ":|."
}

sbIII = \relative c' {
  \key a \minor
  \time 4/4
  \clef bass
  \voiceOne

```



```

\partial 4 a8 b
c4 b a b8 c
d4 d8 c b2
gis4 g g8 a b4
b a8 g f4 e
e2.
}

sbIV = \relative c' {
  \key a \minor
  \time 4/4
  \clef bass
  \voiceTwo

  \partial 4 a,4
  a' gis a e
  f8 e d4 e2
  e,4\fermata e' b a8 g
  c4 f8 e d4 e
  a,2.\fermata
  \bar ":|."
}

simpleBach = \new StaffGroup = "simple Bach" <<
  \new Staff <<
    \new Voice \sbI
    \new Voice \sbII
  >>

  \new Lyrics \txt

  \new Staff <<
    \new Voice \sbIII
    \new Voice \sbIV
  >>
>>

% Third StaffGroup "chromatic Bach"

cbI = \relative c' {
  \textMark \markup {
    \bold "Teacher:"
    \column {
      "Well, you simply copied and transposed a version of J.S.Bach."
      "Do you know this one?"
    }
  }
}

\key a \minor
\time 4/4
\voiceOne

```

```

\partial 4 e4
a b c b
a b gis4. fis8
e4\fermata g! g f
e a a8 b gis4
a2.\fermata
\bar " : | ."
}

cbII = \relative c' {
  \key a \minor
  \time 4/4
  \voiceTwo

  \partial 4 c8 d
  e4 e e8 fis gis4
  a8 g! f!4 e2
  b4 e e d
  d8[ cis] d dis e fis e4
  e2.
  \bar " : | ."
}

cbIII = \relative c' {
  \key a \minor
  \time 4/4
  \clef bass
  \voiceOne

  \partial 4 a8 b
  c[ b] a gis8 a4 d,
  e8[ e'] d c b4. a8
  gis4 b c d8 c
  b[ a] a b c b b c16 d
  c2.
}

cbIV = \relative c' {
  \key a \minor
  \time 4/4
  \clef bass
  \voiceTwo

  \partial 4 a4
  c, e a, b
  c d e2
  e4\fermata e a b8 c
  gis[ g] fis f e dis e4
  a,2.\fermata
  \bar " : | ."
}

```

```

chromaticBach = \new StaffGroup = "chromatic Bach" <<
  \new Staff <<
    \new Voice \cbI
    \new Voice \cbII
  >>

  \new Lyrics \txt

  \new Staff <<
    \new Voice \cbIII
    \new Voice \cbIV
  >>
>>

% Score

\score {
  <<
    \exercise
    \simpleBach
    \chromaticBach
  >>

  \header {
    title = \markup \column {
      \combine \null \vspace #1
      "Exercise: Improve the given choral"
      " "
    }
  }

  \layout {
    \context {
      \Lyrics
      \override LyricText.X-offset = #-1
    }
  }
}

```

### Exercise: Improve the given choral

**Teacher:** This is a simple setting of the choral. Please improve it.

Wer nur den lie - ben Gott läßt wal - ten und hof-fet auf ihn al - le Zeit

**Pupil:** Here's my version!

Wer nur den lie - ben Gott läßt wal - ten und hof-fet auf ihn al - le Zeit

**Teacher:** Well, you simply copied and transposed a version of J.S.Bach.  
Do you know this one?

Wer nur den lie - ben Gott läßt wal - ten und hof-fet auf ihn al - le Zeit

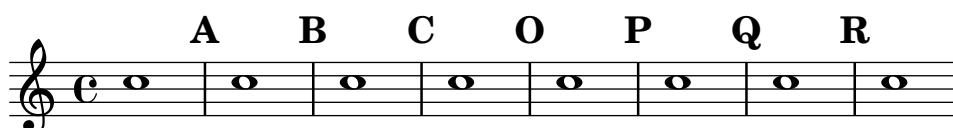
## 28 Preparing parts

### Forcing rehearsal marks to start from a given letter or number

This snippet demonstrates how to obtain automatic ordered rehearsal marks, but from the letter or number desired.

```
\relative c' ' {
  \override Score.RehearsalMark.Y-offset = #3.5

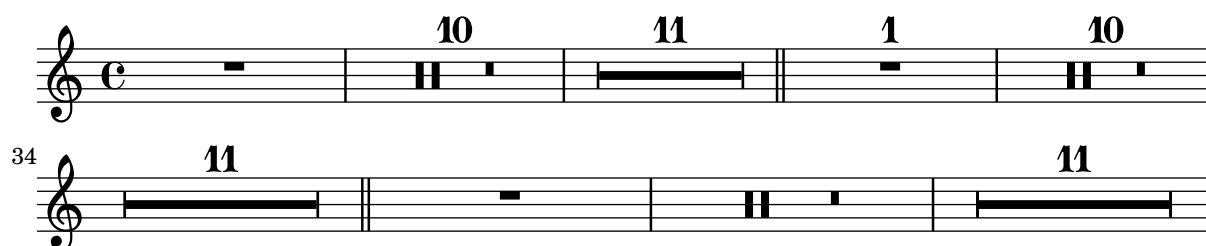
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1 \mark #14
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1
}
```



### Numbering single measure rests

Multi-measure rests show their length by a number except for single measures. This can be changed by setting `restNumberThreshold`.

```
{
  \compressEmptyMeasures
  R1 R1*10 R1*11 \bar "||"
  \set restNumberThreshold = 0
  R1 R1*10 R1*11 \bar "||"
  \set restNumberThreshold = 10
  R1 R1*10 R1*11
}
```



### String quartet template with separate parts

The “String quartet template (simple)” snippet produces a nice string quartet, but what if you need to print parts? This new template demonstrates how to use the `\tag` feature to easily split a piece into individual parts.

For technical reasons, multiple output files cannot be shown here for a single snippet, which means that the template below unifies the code for separate files. The file names are contained in comments at the beginning of each file.

piece.ly contains all the music definitions. The other files – score.ly, vn1.ly, vn2.ly, vla.ly, and vlc.ly – produce the full score and the four parts.

Do not forget to remove specified comments when using separate files!

```
% piece.ly
% (This is the global definitions file.)

global= {
  \time 4/4
  \key c \major
}

Violinone = \new Voice \relative c' {
  c2 d e1
  \bar "|"
}

Violintwo = \new Voice \relative c' {
  g2 g e1
  \bar "|"
}

Viola = \new Voice \relative c' {
  \clef alto
  e2 d c1
  \bar "|"
}

Cello = \new Voice \relative c' {
  \clef bass
  c2 b a1
  \bar "|"
}

music = <<
  \tag #'score \tag #'vn1
  \new Staff \with { instrumentName = "Violin 1" }
    << \global \Violinone >>

  \tag #'score \tag #'vn2
  \new Staff \with { instrumentName = "Violin 2" }
    << \global \Violintwo >>

  \tag #'score \tag #'vla
  \new Staff \with { instrumentName = "Viola" }
    << \global \Viola >>

  \tag #'score \tag #'vlc
  \new Staff \with { instrumentName = "Cello" }
    << \global \Cello >>
>>
```

```
% These are the other files you need to save on your computer
```

```
% score.ly
```

```
% (This is the main file.)
```

```
% Uncomment the line below when using a separate file.
```

```
% \include "piece.ly"
```

```
#(set-global-staff-size 14)
```

```
\score {
```

```
  \new StaffGroup \keepWithTag #'score \music
```

```
  \layout { }
```

```
  \midi { }
```

```
}
```

```
%{ Uncomment this block when using separate files.
```

```
% vn1.ly
```

```
% (This is the Violin 1 part file.)
```

```
\include "piece.ly"
```

```
\score {
```

```
  \keepWithTag #'vn1 \music
```

```
  \layout { }
```

```
}
```

```
% vn2.ly
```

```
% (This is the Violin 2 part file.)
```

```
\include "piece.ly"
```

```
\score {
```

```
  \keepWithTag #'vn2 \music
```

```
  \layout { }
```

```
}
```

```
% vla.ly
```

```
% (This is the Viola part file.)
```

```
\include "piece.ly"
```

```
\score {
```

```
  \keepWithTag #'vla \music
```

```
  \layout { }
```

```
}
```

```
% vlc.ly
```

```
% (This is the Cello part file.)
```

```
\include "piece.ly"
```

```
\score {
```

```
\keepWithTag #'vlc \music  
\layout { }  
}  
  
%}
```

Violin 1

Violin 2

Viola

Cello

The image shows a musical score for four instruments: Violin 1, Violin 2, Viola, and Cello. The score is written in common time (C) and consists of two measures. Violin 1 plays a half note G4, a half note A4, and a whole note B4. Violin 2 plays a half note E4, a half note F4, and a whole note G4. Viola plays a half note C3, a half note D3, and a whole note E3. Cello plays a half note C2, a half note D2, and a whole note E2. The score is enclosed in a large brace on the left.



## 29 Real music

### Changing MIDI output to one channel per voice

When outputting MIDI, the default behavior is for each staff to represent one MIDI channel, with all the voices on a staff amalgamated. This minimizes the risk of running out of MIDI channels, since there are only 16 available per track.

However, by moving the `Staff_performer` to the `Voice` context, each voice on a staff can have its own MIDI channel, as is demonstrated by the following example: despite being on the same staff, two MIDI channels are created, each with a different `midiInstrument`.

```
\score {
  \new Staff <<
    \new Voice \relative c'' {
      \set midiInstrument = "flute"
      \voiceOne
      \key g \major
      \time 2/2
      r2 g-"Flute" ~
      g fis ~
      fis4 g8 fis e2 ~
      e4 d8 cis d2
    }
    \new Voice \relative c'' {
      \set midiInstrument = "clarinet"
      \voiceTwo
      b1-"Clarinet"
      a2. b8 a
      g2. fis8 e
      fis2 r
    }
  >>
  \layout { }
  \midi {
    \context {
      \Staff
      \remove "Staff_performer"
    }
    \context {
      \Voice
      \consists "Staff_performer"
    }
    \tempo 2 = 72
  }
}
```



## Creating a sequence of notes on various pitches

In music that contains many occurrences of the same sequence of notes at different pitches, the following music function may prove useful. It takes a note, of which only the pitch is used.

This example creates the rhythm used throughout *Mars*, from Gustav Holst's *The Planets*.

```
rhythm =
#(define-music-function (p) (ly:pitch?)
  "Make the rhythm in Mars (the Planets) at the given pitch"
  #{ \tuplet 3/2 { $p 8 8 8 } 4 4 8 8 4 #})

\new Staff {
  \time 5/4
  \rhythm c'
  \rhythm c''
  \rhythm g
}
```



## Creating slurs across voices

In some situations it is necessary to create slurs between notes from different voices. The solution is to add invisible notes to one of the voices, using `\hideNotes`.

This example is measure 235 of the Ciaccona from Bach's second partita for solo violin, BWV 1004.

```
\relative c' {
  <<
  {
    d16( a') s a s a[ s a] s a[ s a]
  }
  \\\
  {
    \slurUp
    bes,16[ s e](
    \hideNotes a)
    \unHideNotes f[(
    \hideNotes a)
    \unHideNotes fis](
    \hideNotes a)
    \unHideNotes g[(
    \hideNotes a)
    \unHideNotes gis](
    \hideNotes a)
  }
  >>
}
```

## 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
  }
}
>>
```

# Demo of MIDI instruments

Problem: How to know which midiInstrument values would be best for your composition?

Solution: A LilyPond demo file. You have to compile this snippet by yourself and listen to the created MIDI output file.

```
melody = \relative c' {
  \tempo 4 = 150
  c4.\mf g c16 b' c d
  e16 d e f g4 g'4 r
  R1
}
```

\score {

```

\new Voice \melody
\layout { }
}

\score {
  \new Voice {
    r\mf
    % 1-8 keyboard
    \set Staff.midiInstrument = "acoustic grand" \melody
    \set Staff.midiInstrument = "bright acoustic" \melody
    \set Staff.midiInstrument = "electric grand" \melody
    \set Staff.midiInstrument = "honky-tonk" \melody
    \set Staff.midiInstrument = "electric piano 1" \melody
    \set Staff.midiInstrument = "electric piano 2" \melody
    \set Staff.midiInstrument = "harpsichord" \melody
    \set Staff.midiInstrument = "clav" \melody

    % 9-16 chrom percussion
    \set Staff.midiInstrument = "celesta" \melody
    \set Staff.midiInstrument = "glockenspiel" \melody
    \set Staff.midiInstrument = "music box" \melody
    \set Staff.midiInstrument = "vibraphone" \melody
    \set Staff.midiInstrument = "marimba" \melody
    \set Staff.midiInstrument = "xylophone" \melody
    \set Staff.midiInstrument = "tubular bells" \melody
    \set Staff.midiInstrument = "dulcimer" \melody

    % 17-24 organ
    \set Staff.midiInstrument = "drawbar organ" \melody
    \set Staff.midiInstrument = "percussive organ" \melody
    \set Staff.midiInstrument = "rock organ" \melody
    \set Staff.midiInstrument = "church organ" \melody
    \set Staff.midiInstrument = "reed organ" \melody
    \set Staff.midiInstrument = "accordion" \melody
    \set Staff.midiInstrument = "harmonica" \melody
    \set Staff.midiInstrument = "concertina" \melody

    % 25-32 guitar
    \set Staff.midiInstrument = "acoustic guitar (nylon)" \melody
    \set Staff.midiInstrument = "acoustic guitar (steel)" \melody
    \set Staff.midiInstrument = "electric guitar (jazz)" \melody
    \set Staff.midiInstrument = "electric guitar (clean)" \melody
    \set Staff.midiInstrument = "electric guitar (muted)" \melody
    \set Staff.midiInstrument = "overdriven guitar" \melody
    \set Staff.midiInstrument = "distorted guitar" \melody
    \set Staff.midiInstrument = "guitar harmonics" \melody

    % 33-40 bass
    \set Staff.midiInstrument = "acoustic bass" \melody
    \set Staff.midiInstrument = "electric bass (finger)" \melody
    \set Staff.midiInstrument = "electric bass (pick)" \melody
    \set Staff.midiInstrument = "fretless bass" \melody
  }
}

```

```

\set Staff.midiInstrument = "slap bass 1" \melody
\set Staff.midiInstrument = "slap bass 2" \melody
\set Staff.midiInstrument = "synth bass 1" \melody
\set Staff.midiInstrument = "synth bass 2" \melody

% 41-48 strings
\set Staff.midiInstrument = "violin" \melody
\set Staff.midiInstrument = "viola" \melody
\set Staff.midiInstrument = "cello" \melody
\set Staff.midiInstrument = "contrabass" \melody
\set Staff.midiInstrument = "tremolo strings" \melody
\set Staff.midiInstrument = "pizzicato strings" \melody
\set Staff.midiInstrument = "orchestral harp" \melody
\set Staff.midiInstrument = "timpani" \melody

% 49-56 ensemble
\set Staff.midiInstrument = "string ensemble 1" \melody
\set Staff.midiInstrument = "string ensemble 2" \melody
\set Staff.midiInstrument = "synthstrings 1" \melody
\set Staff.midiInstrument = "synthstrings 2" \melody
\set Staff.midiInstrument = "choir aahs" \melody
\set Staff.midiInstrument = "voice oohs" \melody
\set Staff.midiInstrument = "synth voice" \melody
\set Staff.midiInstrument = "orchestra hit" \melody

% 57-64 brass
\set Staff.midiInstrument = "trumpet" \melody
\set Staff.midiInstrument = "trombone" \melody
\set Staff.midiInstrument = "tuba" \melody
\set Staff.midiInstrument = "muted trumpet" \melody
\set Staff.midiInstrument = "french horn" \melody
\set Staff.midiInstrument = "brass section" \melody
\set Staff.midiInstrument = "synthbrass 1" \melody
\set Staff.midiInstrument = "synthbrass 2" \melody

% 65-72 reed
\set Staff.midiInstrument = "soprano sax" \melody
\set Staff.midiInstrument = "alto sax" \melody
\set Staff.midiInstrument = "tenor sax" \melody
\set Staff.midiInstrument = "baritone sax" \melody
\set Staff.midiInstrument = "oboe" \melody
\set Staff.midiInstrument = "english horn" \melody
\set Staff.midiInstrument = "bassoon" \melody
\set Staff.midiInstrument = "clarinet" \melody

% 73-80 pipe
\set Staff.midiInstrument = "piccolo" \melody
\set Staff.midiInstrument = "flute" \melody
\set Staff.midiInstrument = "recorder" \melody
\set Staff.midiInstrument = "pan flute" \melody
\set Staff.midiInstrument = "blown bottle" \melody
\set Staff.midiInstrument = "shakuhachi" \melody

```

```

\set Staff.midiInstrument = "whistle" \melody
\set Staff.midiInstrument = "ocarina" \melody

% 81-88 synth lead
\set Staff.midiInstrument = "lead 1 (square)" \melody
\set Staff.midiInstrument = "lead 2 (sawtooth)" \melody
\set Staff.midiInstrument = "lead 3 (calliope)" \melody
\set Staff.midiInstrument = "lead 4 (chiff)" \melody
\set Staff.midiInstrument = "lead 5 (charang)" \melody
\set Staff.midiInstrument = "lead 6 (voice)" \melody
\set Staff.midiInstrument = "lead 7 (fifths)" \melody
\set Staff.midiInstrument = "lead 8 (bass+lead)" \melody

% 89-96 synth pad
\set Staff.midiInstrument = "pad 1 (new age)" \melody
\set Staff.midiInstrument = "pad 2 (warm)" \melody
\set Staff.midiInstrument = "pad 3 (polysynth)" \melody
\set Staff.midiInstrument = "pad 4 (choir)" \melody
\set Staff.midiInstrument = "pad 5 (bowed)" \melody
\set Staff.midiInstrument = "pad 6 (metallic)" \melody
\set Staff.midiInstrument = "pad 7 (halo)" \melody
\set Staff.midiInstrument = "pad 8 (sweep)" \melody

% 97-104 synth effects
\set Staff.midiInstrument = "fx 1 (rain)" \melody
\set Staff.midiInstrument = "fx 2 (soundtrack)" \melody
\set Staff.midiInstrument = "fx 3 (crystal)" \melody
\set Staff.midiInstrument = "fx 4 (atmosphere)" \melody
\set Staff.midiInstrument = "fx 5 (brightness)" \melody
\set Staff.midiInstrument = "fx 6 (goblins)" \melody
\set Staff.midiInstrument = "fx 7 (echoes)" \melody
\set Staff.midiInstrument = "fx 8 (sci-fi)" \melody

% 105-112 ethnic
\set Staff.midiInstrument = "sitar" \melody
\set Staff.midiInstrument = "banjo" \melody
\set Staff.midiInstrument = "shamisen" \melody
\set Staff.midiInstrument = "koto" \melody
\set Staff.midiInstrument = "kalimba" \melody
\set Staff.midiInstrument = "bagpipe" \melody
\set Staff.midiInstrument = "fiddle" \melody
\set Staff.midiInstrument = "shanai" \melody

% 113-120 percussive
\set Staff.midiInstrument = "tinkle bell" \melody
\set Staff.midiInstrument = "agogo" \melody
\set Staff.midiInstrument = "steel drums" \melody
\set Staff.midiInstrument = "woodblock" \melody
\set Staff.midiInstrument = "taiko drum" \melody
\set Staff.midiInstrument = "melodic tom" \melody
\set Staff.midiInstrument = "synth drum" \melody
\set Staff.midiInstrument = "reverse cymbal" \melody

```

```

% 121-128 sound effects
\set Staff.midiInstrument = "guitar fret noise" \melody
\set Staff.midiInstrument = "breath noise" \melody
\set Staff.midiInstrument = "seashore" \melody
\set Staff.midiInstrument = "bird tweet" \melody
\set Staff.midiInstrument = "telephone ring" \melody
\set Staff.midiInstrument = "helicopter" \melody
\set Staff.midiInstrument = "applause" \melody
\set Staff.midiInstrument = "gunshot" \melody
}
\midi { }
}

```



## Dotted harmonics

Artificial harmonics using `\harmonic` do not show dots. To override this behavior, set the context property `harmonicDots`.

```

\relative c' {} {
  \time 3/4
  \key f \major
  \set harmonicDots = ##t
  <bes f'\harmonic>2. ~
  <bes f'\harmonic>4. <a e'\harmonic>8( <gis dis'\harmonic> <g d'\harmonic>)
  <fis cis'\harmonic>2.
  <bes f'\harmonic>2.
}

```



## Heavily customized polymetric time signatures

Though the polymetric time signature shown is not the most essential item here, it has been included to show the beat of this piece (which is the template of a real Balkan song, by the way).

```

melody = \relative c' {} {
  \key g \major
  \compoundMeter #'((3 8) (2 8) (2 8) (3 8) (2 8) (2 8)
                    (2 8) (2 8) (3 8) (2 8) (2 8))
  \set Timing.beamExceptions = #'()
  \set Timing.beatStructure = 3,2,2,3,2,2,2,2,3,2,2
  c8 c c d4 c8 c b c b a4 g fis8 e d c b' c d e4-~ fis8 g \break
  c,4. d4 c4 d4. c4 d c2 d4. e4-~ d4
  c4. d4 c4 d4. c4 d c2 d4. e4-~ d4 \break
}

```

```

}

drum = \new DrumStaff \drummode {
  \repeat volta 2 {
    bd4.^{\markup { Drums } sn4 bd \bar "}
    sn4. bd4 sn \bar "
    bd sn bd4. sn4 bd
  }
}

```

```

\new Staff {
  \melody
  \drum
}

```

## 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.

The following example typesets measure 65 of Debussy's prelude *Les collines d'Anacapri*.

```

\new PianoStaff <<
  \set PianoStaff.connectArpeggios = ##t
  \override PianoStaff.Arpeggio.stencil =
    #ly:arpeggio::brew-chord-bracket

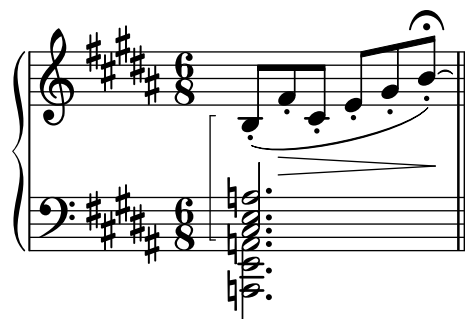
  \new Staff \relative c' {
    \key b \major
    \time 6/8
    b8-.(\arpeggio fis'-.\> cis-.
      e-. gis-. b-.)\!\fermata^\laissezVibrer \bar "||"
  }

  \new Staff \relative c' {
    \clef bass
    \key b \major
    << { <a e cis>2.\arpeggio } \
      { <a, e a,>2. } >>
  }

```



```
}
>>
```



## Inserting score fragments above a staff, as markups

The `\markup` command is quite versatile. In this snippet, it contains a `\score` block instead of texts or marks.

```
tuning = \markup \score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \clef bass
    <c, g, d g>1
  }
  \layout {
    indent = 0\cm
  }
}

\header {
  title = "Solo Cello Suites"
  subtitle = "Suite IV"
  subsubtitle = \markup { Originalstimmung: \raise #0.5 \tuning }
  tagline = ##f
}

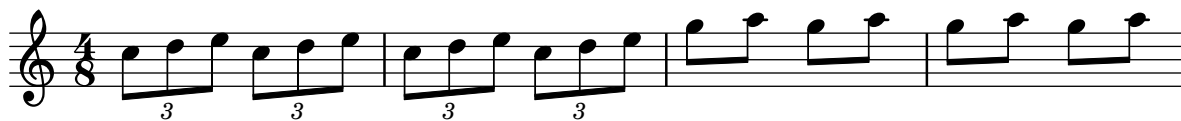
\layout {
  ragged-right = ##f
}

\relative c'' {
  \time 4/8
  \tuplet 3/2 { c8 d e } \tuplet 3/2 { c d e }
  \tuplet 3/2 { c8 d e } \tuplet 3/2 { c d e }
  g8 a g a
  g8 a g a
}
```

## Solo Cello Suites

### Suite IV

Originalstimmung: 



## Percussion example

A short example taken from Stravinsky's *L'histoire du Soldat*.

```
#(define mydrums '((bassdrum   default #f  4)
                   (snare      default #f -4)
                   (tambourine default #f  0)))
```

```
U = \stemUp
D = \stemDown
```

```
global = {
  \time 3/8 s4.
  \time 2/4 s2*2
  \time 3/8 s4.
  \time 2/4 s2
}
```

```
drumsA = {
  \context DrumVoice <<
    \global
    \drummode {
      \autoBeamOff
      \D sn8 \U tamb s |
      sn4 \D sn4 |
      \U tamb8 \D sn \U sn16 \D sn \U sn8 |
      \D sn8 \U tamb s |
      \U sn4 s8 \U tamb
    }
  >>
}
```

```
drumsB = \drummode {
  s4 bd8 s2*2 s4 bd8 s4 bd8 s
}
```

```
\layout {
  indent = 40\mm
  \context {
    \DrumStaff
    drumStyleTable = #(alist->hash-table mydrums)
  }
}
```

```
\score {
  \new StaffGroup <<
    \new DrumStaff \with {
      instrumentName = \markup \center-column {
        "Tambourine"
      }
    }
  >>
}
```

```

    "et"
    "caisse claire s. timbre" }
} \drumsA
\new DrumStaff \with {
    instrumentName = "Grosse Caisse"
}\drumsB
>>
}

```

Tambourine  
et  
caisse claire s. timbre

Grosse Caisse

## Printing music with different time signatures

In the following snippet, two parts have a completely different time signature, yet remain synchronized.

The bar lines can no longer be printed at the Score level; to allow independent bar lines in each part, the `Default_barline_engraver` and `Timing_translator` are moved from the Score context to the Staff context.

If bar numbers are required, the `Bar_number_engraver` should also be moved, since it relies on properties set by the `Timing_translator`; a `\with` block can be used to add bar numbers to the relevant staff.

```

global = {
    \time 3/4 s2.*3 \break
    s2.*3
}

\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Bar_number_engraver"
    \override SpacingSpanner.uniform-stretching = ##t
    \override SpacingSpanner.strict-note-spacing = ##t
    \proportionalNotationDuration = #1/64
  }
  \context {
    \Staff
    \consists "Timing_translator"
  }
  \context {
    \Voice
    \remove "Forbid_line_break_engraver"
    \tupletFullLength = ##t
  }
}

```

```

Bassklarinette = \new Staff \with {
  \consists "Bar_number_engraver"
  barNumberVisibility = #(every-nth-bar-number-visible 2)
  \override BarNumber.break-visibility = #end-of-line-invisible
} <<
\global
{
  \clef treble
  \time 3/8 d''4. |
  \time 3/4 r8 des''2( c''8) |
  \time 7/8 r4. ees''2 ~ |
  \time 2/4 \tupletUp \tuplet 3/2 { ees''4 r4 d''4 ~ } |
  \time 3/8 \tupletUp \tuplet 4/3 { d''4 r4 } |
  \time 2/4 e''2 |
  \time 3/8 es''4. |
  \time 3/4 r8 d''2 r8 |
}
>>

Perkussion = \new StaffGroup <<
  \new Staff <<
    \global
    {
      \clef percussion
      \time 3/4 r4 c'2 ~ |
      c'2. |
      R2. |
      r2 g'4 ~ |
      g'2. ~ |
      g'2. |
    }
  >>
  \new Staff <<
    \global {
      \clef percussion
      \time 3/4 R2. |
      g'2. ~ |
      g'2. |
      r4 g'2 ~ |
      g'2 r4 |
      g'2. |
    }
  >>
>>

\score {
  <<
    \Bassklarinette
    \Perkussion
  >>
}

```

The musical score is written for a single melodic line on a treble staff and a piano accompaniment on a grand staff (bass and tenor staves). The key signature has one flat (B-flat), and the time signature is 3/4. The score is divided into three systems, with the first system starting at measure 1 and the third system starting at measure 8.

**System 1 (Measures 1-3):**

- Measure 1:** Treble staff has a dotted quarter note (B4). Bass staff has a whole rest.
- Measure 2:** Treble staff has a quarter rest, followed by a dotted half note (B-flat4) with a finger number '2' above it. Bass staff has a dotted half note (B-flat4).
- Measure 3:** Treble staff has a quarter note (A4) with a finger number '4' above it, followed by a quarter rest. Bass staff has a dotted half note (B-flat4).

**System 2 (Measures 4-6):**

- Measure 4:** Treble staff has a quarter note (G4) with a finger number '3' above it, followed by a quarter rest. Bass staff has a dotted half note (B-flat4).
- Measure 5:** Treble staff has a quarter note (F4) with a finger number '4' above it, followed by a quarter rest. Bass staff has a dotted half note (B-flat4).
- Measure 6:** Treble staff has a quarter note (E4) with a finger number '6' above it, followed by a quarter rest. Bass staff has a dotted half note (B-flat4).

**System 3 (Measures 7-8):**

- Measure 7:** Treble staff has a quarter note (D4) with a finger number '3' above it, followed by a quarter rest. Bass staff has a dotted half note (B-flat4).
- Measure 8:** Treble staff has a quarter note (C4) with a finger number '3' above it, followed by a quarter rest. Bass staff has a dotted half note (B-flat4).

## 30 Really cool

### Adding the current date to a score

With a little Scheme code, the current date can easily be added to a score.

```
\paper { tagline = ##f }

% first, define a variable to hold the formatted date:
date = #(strftime "%d-%m-%Y" (localtime (current-time)))

% use it in the title block:
\header {
  title = "Including the date!"
  subtitle = \date
}

\score {
  \relative c' {
    c4 c c c
  }
}

% and use it in a \markup block:
\markup {
  \date
}
```

### Including the date!

17-01-2026



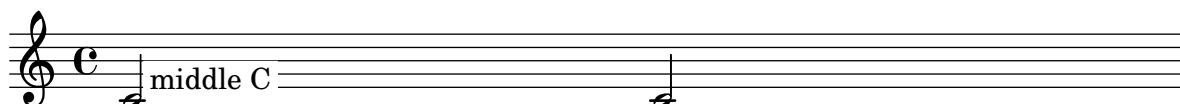
17-01-2026

### Blanking staff lines using the \whiteout command

The \whiteout command underlays a markup with a white box. Since staff lines are in a lower layer than most other grobs, this white box will not overlap any other grob.

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

\relative c' {
  \override TextScript.extra-offset = #'(2 . 4)
  c2-\markup { \whiteout \pad-markup #0.5 "middle C" } c
}
```





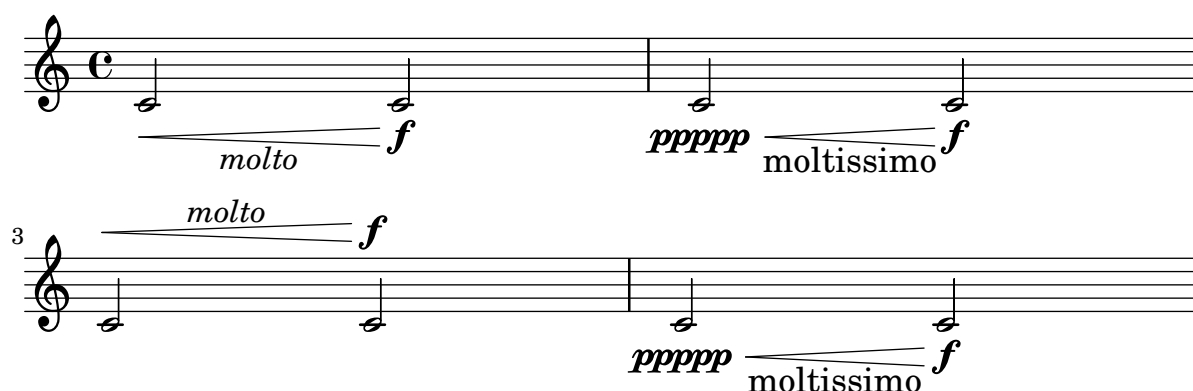
```

x-shift X))))
#})

hairpinMolto = \hairpinWithCenteredText \markup { \italic molto }
hairpinMore = \hairpinWithCenteredText \markup { \larger moltissimo }

\relative c' {
  \hairpinMolto c2\< c\f
  \hairpinMore c2\ppppp\< c\f
  \break
  \hairpinMolto c2^\< c\f
  \hairpinMore c2\ppppp\< c\f
}

```



## Changing properties for individual grobs

The `\applyOutput` command allows the tuning of any layout object, in any context. It requires a Scheme function with three arguments.

In the example below, function `mc-squared` is executed for all `NoteHead` grobs (within the current `Voice` context) at the current time step; the function modifies the grob's stencil, using the `staff-position` property to replace some pitches with markup.

See the 'Extending' manual (<https://lilypond.org/doc/v2.24/Documentation/extending/running-a-function-on-all-layout-objects>) for more information.

```

#(define (mc-squared grob grob-origin context)
  (let ((sp (ly:grob-property grob 'staff-position)))
    (ly:grob-set-property!
     grob 'stencil
     (grob-interpret-markup grob
      #{ \markup \lower #0.5
        #(case sp
            ((-5) "m")
            ((-3) "c ")
            ((-2) #{ \markup \teeny \bold 2 #})
            (else "bla")) #}))))

```

```

\relative c' {
  <d f g b>2
  \applyOutput Voice.NoteHead #mc-squared
  <d f g b>2
}

```





## Clusters

Clusters are a device to denote that a complete range of notes is to be played.

```
fragment = \relative c' {
  c4 f <e d'>4
  <g a>8 <e a> a4 c2 <d b>4
  e2 c
}

<<
  \new Staff \fragment
  \new Staff \makeClusters \fragment
>>
```



## Coloring notes depending on their pitch

It is possible to color note heads depending on their pitch and/or their names: the function used in this example even makes it possible to distinguish enharmonics.

```
% Association list of pitches to colors.
#(define color-mapping
  (list
    (cons (ly:make-pitch 0 0 NATURAL) (x11-color 'red))
    (cons (ly:make-pitch 0 0 SHARP) (x11-color 'green))
    (cons (ly:make-pitch 0 1 FLAT) (x11-color 'green))
    (cons (ly:make-pitch 0 2 NATURAL) (x11-color 'red))
    (cons (ly:make-pitch 0 2 SHARP) (x11-color 'green))
    (cons (ly:make-pitch 0 3 FLAT) (x11-color 'red))
    (cons (ly:make-pitch 0 3 NATURAL) (x11-color 'green))
    (cons (ly:make-pitch 0 4 SHARP) (x11-color 'red))
    (cons (ly:make-pitch 0 5 NATURAL) (x11-color 'green))
    (cons (ly:make-pitch 0 5 FLAT) (x11-color 'red))
    (cons (ly:make-pitch 0 6 SHARP) (x11-color 'red))
    (cons (ly:make-pitch 0 1 NATURAL) (x11-color 'blue))
    (cons (ly:make-pitch 0 3 SHARP) (x11-color 'blue))
    (cons (ly:make-pitch 0 4 FLAT) (x11-color 'blue))
    (cons (ly:make-pitch 0 5 SHARP) (x11-color 'blue))
    (cons (ly:make-pitch 0 6 FLAT) (x11-color 'blue))))

% Compare pitch and alteration (not octave).
#(define (pitch-equals? p1 p2)
  (and
```

```

(= (ly:pitch-alteration p1) (ly:pitch-alteration p2))
(= (ly:pitch-notename p1) (ly:pitch-notename p2))))

#(define (pitch-to-color pitch)
  (let ((color (assoc pitch color-mapping pitch-equals?)))
    (if color
      (cdr color))))

#(define (color-notehead grob)
  (pitch-to-color
    (ly:event-property (event-cause grob) 'pitch)))

\score {
  \new Staff \relative c' {
    \override NoteHead.color = #color-notehead
    c8 b d dis ees f g aes
  }
}

```



## Creating a sequence of notes on various pitches

In music that contains many occurrences of the same sequence of notes at different pitches, the following music function may prove useful. It takes a note, of which only the pitch is used.

This example creates the rhythm used throughout *Mars*, from Gustav Holst's *The Planets*.

```

rhythm =
#(define-music-function (p) (ly:pitch?)
  "Make the rhythm in Mars (the Planets) at the given pitch"
  #{ \tuplet 3/2 { $p 8 8 8 } 4 4 8 8 4 #})

\new Staff {
  \time 5/4
  \rhythm c'
  \rhythm c''
  \rhythm g
}

```



## Generating random notes

This Scheme-based snippet generates random notes. Use as

```
\randomNotes n from to dur
```

to generate *n* random notes between pitches *from* and *to*, with duration *dur*.

```

randomNotes =
#(define-music-function (n from to dur)
  (integer? ly:pitch? ly:pitch? ly:duration?))

```

```
(let ((from-step (ly:pitch-steps from))
      (to-step (ly:pitch-steps to)))
  (make-sequential-music
    (map (lambda (_)
          (let* ((step (+ from-step
                          (random (- to-step from-step))))
                (pitch (ly:make-pitch 0 step 0)))
            #{ $pitch $dur #}))
      (iota n))))
```

```
\randomNotes 24 c' g'' 8
```



## Generating whole scores (also book parts) in Scheme without using the parser

A LilyPond score internally is just a Scheme expression, generated by the LilyPond parser. Using Scheme, one can also automatically generate a score without an input file. If you have the music expression in Scheme, a score can be generated by simply calling

```
(scorify-music music)
```

on your music. This generates a score object, for which you can then set a custom layout block with

```
(let* ((layout (ly:output-def-clone $defaultlayout)))
  ; modify the layout here, then assign it:
  (ly:score-add-output-def! score layout))
```

Finally, all you have to do it to pass this score to LilyPond for typesetting. This snippet defines functions (add-score score), (add-text text), and (add-music music) to pass a complete score, some markup, or some music to LilyPond for typesetting.

This snippet also works for typesetting scores inside a `\book {...}` block as well as top-level scores. To achieve this, each score scheduled for typesetting is appended to the list of top-level scores, and the top-level book handler (which is a Scheme function called to process a book once a `\book{...}` block is closed) is modified to insert all collected scores so far to the book.

Note: For technical reasons, only the first `\book` is shown, as the other `\book` commands create additional output files.

```
#(define-public (add-score score)
  (ly:parser-define! 'toplevel-scores
    (cons score (ly:parser-lookup 'toplevel-scores))))

#(define-public (add-text text)
  (add-score (list text)))

#(define-public (add-music music)
  (collect-music-aux (lambda (score)
    (add-score score))
    music))

#(define-public (toplevel-book-handler book)
```

```

    (map (lambda (score)
          (ly:book-add-score! book score))
         (reverse! (ly:parser-lookup 'toplevel-scores)))
    (ly:parser-define! 'toplevel-scores (list))
    (print-book-with-defaults book))

#(define-public (book-score-handler book score)
  (add-score score))

#(define-public (book-text-handler book text)
  (add-text text))

#(define-public (book-music-handler book music)
  (add-music music))

% Some example code to show how to use these functions. Each call to
% `oneNoteScore` constructs a global markup followed by a single
% staff with a single quarter note. The pitch of this note is taken
% from the variable `pitch`; the start value 0 corresponds to pitch C.
% After emitting the score, variable `pitch` gets increased by 1.
%
% `oneNoteScore` calls Scheme function `add-one-note-score` to do all
% the work.

#(define add-one-note-score #f)
#(let ((pitch 0))
  (set! add-one-note-score
    (lambda ()
      (let* ((music
              (make-music
               'EventChord
               'elements (list (make-music
                               'NoteEvent
                               'duration (ly:make-duration 2 0 1/1)
                               'pitch (ly:make-pitch 0 pitch 0)))))
              (score (scorify-music music))
              (layout (ly:output-def-clone $defaultlayout))
              (note-name (case pitch
                          ((0) "do")
                          ((1) "ré")
                          ((2) "mi")
                          ((3) "fa")
                          ((4) "sol")
                          ((5) "la")
                          ((6) "si")
                          (else "huh"))
              (title (markup #:large #:line
                            ("Score with a" note-name))))
              (ly:score-add-output-def! score layout)
              (add-text title)
              (add-score score))

```

```

(set! pitch (modulo (1+ pitch) 7))))))

oneNoteScore =
#(define-void-function () ()
  (add-one-note-score))

\book {
  \oneNoteScore

  \paper { tagline = ##f }
}

\book {
  \oneNoteScore
  \oneNoteScore

  \paper { tagline = ##f }
}

% Top-level scores are also handled correctly.
\oneNoteScore
\oneNoteScore

\paper { tagline = ##f }

```

Score with a do



## 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'
}

```



## Non-traditional key signatures

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

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

```

\set Staff.keyAlterations =
  #`(((octave . step) . alter) ((octave . step) . alter) ...)

```

where, for each element in the list, *octave* specifies the octave (0 being the octave from middle C to the B above), *step* specifies the note within the octave (0 means C and 6 means B), and *alter* is one of SHARP, FLAT, DOUBLE-SHARP, etc., preceded by a comma.

Alternatively, you can use the more concise format (*step . alter*) for each item in the list if the same alterations are used 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.

```
\include "arabic.ly"
```

```
\relative do' {
  \set Staff.keyAlterations = #`((0 . ,SEMI-FLAT)
                                (1 . ,SEMI-FLAT)
                                (2 . ,FLAT)
                                (5 . ,FLAT)
                                (6 . ,SEMI-FLAT))

  % \set Staff.extraNatural = ##f
  re reb \down reb resd
  dod dob dosd \down dob |
  dobsb dods do do |
}
```



## Printing music with different time signatures

In the following snippet, two parts have a completely different time signature, yet remain synchronized.

The bar lines can no longer be printed at the Score level; to allow independent bar lines in each part, the `Default_barline_engraver` and `Timing_translator` are moved from the `Score` context to the `Staff` context.

If bar numbers are required, the `Bar_number_engraver` should also be moved, since it relies on properties set by the `Timing_translator`; a `\with` block can be used to add bar numbers to the relevant staff.

```
global = {
  \time 3/4 s2.*3 \break
  s2.*3
}

\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Bar_number_engraver"
    \override SpacingSpanner.uniform-stretching = ##t
    \override SpacingSpanner.strict-note-spacing = ##t
    \proportionalNotationDuration = #1/64
  }
  \context {
    \Staff
```

```

    \consists "Timing_translator"
  }
  \context {
    \Voice
    \remove "Forbid_line_break_engraver"
    tupletFullLength = ##t
  }
}

Bassklarinette = \new Staff \with {
  \consists "Bar_number_engraver"
  barNumberVisibility = #(every-nth-bar-number-visible 2)
  \override BarNumber.break-visibility = #end-of-line-invisible
} <<
\global
{
  \clef treble
  \time 3/8 d''4. |
  \time 3/4 r8 des''2( c''8) |
  \time 7/8 r4. ees''2 ~ |
  \time 2/4 \tupletUp \tuplet 3/2 { ees''4 r4 d''4 ~ } |
  \time 3/8 \tupletUp \tuplet 4/3 { d''4 r4 } |
  \time 2/4 e''2 |
  \time 3/8 es''4. |
  \time 3/4 r8 d''2 r8 |
}
>>

Perkussion = \new StaffGroup <<
  \new Staff <<
    \global
    {
      \clef percussion
      \time 3/4 r4 c'2 ~ |
      c'2. |
      R2. |
      r2 g'4 ~ |
      g'2. ~ |
      g'2. |
    }
  >>
  \new Staff <<
    \global {
      \clef percussion
      \time 3/4 R2. |
      g'2. ~ |
      g'2. |
      r4 g'2 ~ |
      g'2 r4 |
      g'2. |
    }
  >>

```

&gt;&gt;

```
\score {  
  <<  
    \Bassklarinette  
    \Perkussion  
  >>  
}
```

The image displays a musical score for two instruments: Bass Clarinet and Percussion. The score is organized into three systems, each with a treble clef staff for the Bass Clarinet and a grand staff (two staves) for the Percussion.

**System 1:** The Bass Clarinet staff begins with a 3/8 time signature, followed by a 2/4 time signature. It contains a melodic line with a slur over two measures, a 7/8 time signature, and a 2/4 time signature. The Percussion grand staff has a 3/4 time signature and contains a single note in the first measure, followed by a rest.

**System 2:** The Bass Clarinet staff starts with a 3/8 time signature, followed by a 2/4 time signature, and then a 3/8 time signature. It features a melodic line with a slur over three measures, a 4/4 time signature, and a 3/4 time signature. The Percussion grand staff has a 3/4 time signature and contains a single note in the first measure, followed by a rest.

**System 3:** The Bass Clarinet staff starts with a 3/4 time signature and contains a melodic line with a slur over two measures. The Percussion grand staff has a 3/4 time signature and contains a single note in the first measure, followed by a rest.

Measure numbers 2, 4, and 8 are indicated above the first staff of each system. Measure numbers 3, 4, and 6 are indicated above the first staff of the second system. Measure numbers 3, 4, and 6 are indicated above the first staff of the third system.

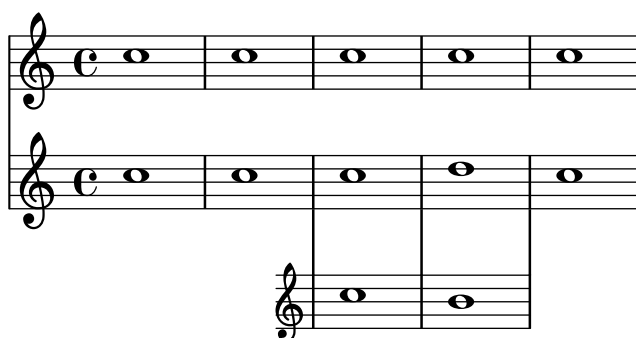


## 31 Really simple

### Adding an extra staff

An extra staff can be added (possibly temporarily) after the start of a piece.

```
\score {
  <<
    \new Staff \relative c'' {
      c1 | c | c | c | c
    }
    \new StaffGroup \relative c'' {
      \new Staff {
        c1 | c
      }
      <<
        { c1 | d }
        \new Staff {
          \once \omit Staff.TimeSignature
          c1 | b
        }
      >>
    }
  >>
}
```



### Adding drum parts

Using the powerful pre-configured tools such as the `\drummode` function and the `DrumStaff` context, inputting drum parts is quite easy: drums are placed at their own staff positions (with a special clef symbol) and have note heads according to the drum. Attaching an extra symbol to the drum or restricting the number of lines is possible.

```
drh = \drummode {
  cymc4.^"crash" hhc16^"h.h." hh hhc8 hho hhc8 hh16 hh
  hhc4 r4 r2
}
dr1 = \drummode {
  bd4 sn8 bd bd4 << bd ss >>
  bd8 tommh tommh bd toml toml bd tomfh16 tomfh
}
timb = \drummode {
```

```

        timh4 ssh timl8 ssh r timh r4
        ssh8 timl r4 cb8 cb
    }

\score {
  <<
    \new DrumStaff \with {
      instrumentName = "timbales"
      drumStyleTable = #timbales-style
      \override StaffSymbol.line-count = #2
      \override BarLine.bar-extent = #'(-1 . 1)
    }
    <<
      \timb
    >>
    \new DrumStaff \with { instrumentName = "drums" }
    <<
      \new DrumVoice { \stemUp \drh }
      \new DrumVoice { \stemDown \drl }
    >>
  >>
  \layout { }
  \midi { \tempo 4 = 120 }
}

```

## Adding fingerings to a score

Fingering instructions can be entered using a simple syntax.

```

\relative c' {
  c4-1 d-2 f-4 e-3
}

```

## Aligning text marks to notes

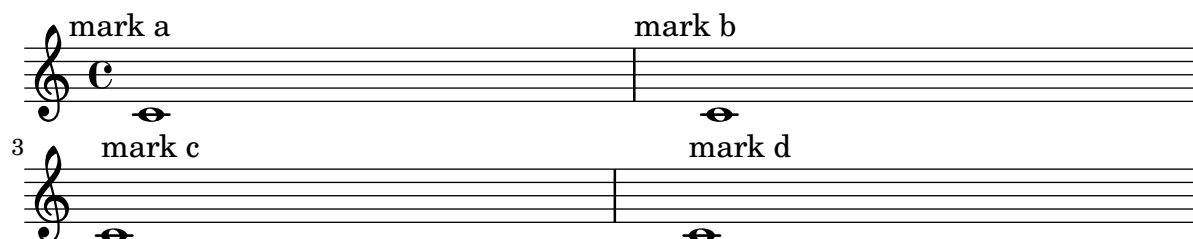
By default, TextMark objects are aligned to so-called NonMusicalPaperColumn grobs, like the left edge of the staff or a bar line. They can be aligned to a note instead by setting the non-musical property to #f.

```

\layout {
  line-length = 80\mm
}

```

```
{
  \textMark "mark a" c'1 |
  \textMark "mark b" c'1 |
  \break
  \override Score.TextMark.non-musical = ##f
  \textMark "mark c" c'1 |
  \textMark "mark d" c'1 |
}
```



## Analysis brackets above the staff

Simple horizontal analysis brackets are added below the staff by default. The following example shows a way to place them above the staff instead.

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

\relative c' {
  \once \override HorizontalBracket.direction = #UP
  c2\startGroup
  d2\stopGroup
}
```

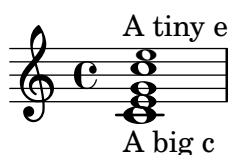


## Changing a single note's size in a chord

Individual note heads in a chord can be modified with the `\tweak` command inside a chord, by altering the `font-size` property.

Inside the chord (within the brackets `< >`), before the note to be altered, place the `\tweak` command, followed by `font-size` and define the proper size like `#-2` (a tiny note head).

```
\relative c' {
  <\tweak font-size #+2 c e g c
  \tweak font-size #-2 e>1
  ^\markup { A tiny e } _\markup { A big c }
}
```



## Changing stanza fonts

Fonts can be changed independently for each stanza, including the font used for printing the stanza number.

```
%{
You may have to install additional fonts.

Red Hat Fedora

    dejavu-fonts-all

Debian GNU/Linux, Ubuntu

    fonts-dejavu-core
    fonts-dejavu-extra
}%

\relative c' ' {
  \time 3/4
  g2 e4
  a2 f4
  g2.
}
\addlyrics {
  \set stanza = #"1. "
  Hi, my name is Bert.
}
\addlyrics {
  \override StanzaNumber.fonts.serif = "DejaVu Sans"
  \set stanza = #"2. "
  \override LyricText.font-family = #'typewriter
  Oh, ché -- ri, je t'aime
}
```



1. Hi, my name is Bert.
2. Oh, ché-ri, je t'aime

## Changing the appearance of a slur from solid to dotted or dashed

The appearance of slurs may be changed from solid to dotted or dashed.

```
\relative c' {
  c4( d e c)
  \slurDotted
  c4( d e c)
  \slurSolid
  c4( d e c)
  \slurDashed
  c4( d e c)
```

```
\slurSolid
c4( d e c)
}
```



## Combining dynamics with markup texts

Some dynamics may involve text indications (such as “*più f*” or “*p subito*”). These can be produced using a `\markup` block; the resulting object behaves like a `TextScript` grob.

See also “Combining dynamics with markup texts (2)”.

```
piuF = \markup { \italic più \dynamic f }
```

```
\markup \with-true-dimensions % work around a cropping issue
\score {
  \relative c'' {
    c2\f c-\piuF
  }
}
```



## Combining dynamics with markup texts (2)

Some dynamics may involve text indications (such as “*più f*” or “*p subito*”). These can be produced using the `make-dynamic-script` Scheme function; the resulting object behaves like a `DynamicText` grob.

See also “Combining dynamics with markup texts”.

```
piuF = #(make-dynamic-script
  #{ \markup { \normal-text \italic più \dynamic f } #})
```

```
\score {
  \relative c'' {
    c2\f c\piuF
  }
}
```



## Display non-English chord names

The default English naming of chords can be changed to other languages, as demonstrated in this snippet.

```
scm = \chordmode {
  c1/c | cis/cis
```

```

    b1/b | bis/bis | bes/bes
}

\layout {
  indent = 3\cm
  ragged-right = ##f

  \context {
    \ChordNames
    \consists "Instrument_name_engraver"
  }
  \context {
    \Score
    \override InstrumentName.self-alignment-Y = -1.2
    \override InstrumentName.self-alignment-X = #RIGHT
  }
}

<<
\new ChordNames {
  \set instrumentName = #"default"
  \scm
}
\new ChordNames {
  \set instrumentName = #"german"
  \germanChords \scm
}
\new ChordNames {
  \set instrumentName = #"semi-german"
  \semiGermanChords \scm
}
\new ChordNames {
  \set instrumentName = #"italian"
  \italianChords \scm
}
\new ChordNames {
  \set instrumentName = #"french"
  \frenchChords \scm
}
\context Voice { \scm }
>>

```

default	C/C	C#/C#	B/B	B#/B#	Bb/Bb
german	C/c	C#/cis	H/h	H#/his	B/b
semi-german	C/c	C#/cis	H/h	H#/his	Bb/b
italian	Do/Do	Do #/Do #	Si/Si	Si #/Si #	Si b/Si b
french	Do/Do	Do #/Do #	Si/Si	Si #/Si #	Si b/Si b

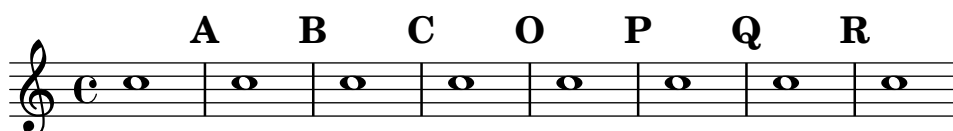


## Forcing rehearsal marks to start from a given letter or number

This snippet demonstrates how to obtain automatic ordered rehearsal marks, but from the letter or number desired.

```
\relative c'' {
  \override Score.RehearsalMark.Y-offset = #3.5

  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1 \mark #14
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1
}
```



## Lyrics alignment

Horizontal alignment for lyrics can be set by overriding the `self-alignment-X` property of the `LyricText` object. Value `-1` means left-aligned, `0` centered, and `1` right-aligned. Alternatively, you can use the Scheme values `LEFT`, `CENTER`, and `RIGHT` instead of numbers. Other numeric values are possible, too – don't forget to add the '#' Scheme prefix for negative numbers!

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

\relative c'' {
  c1 c c c
}

\addlyrics {
  \once \override LyricText.self-alignment-X = #LEFT
  "left-aligned"
  \once \override LyricText.self-alignment-X = #CENTER
  "centered"
```

```

\once \override LyricText.self-alignment-X = 1
"right-aligned"
\once \override LyricText.self-alignment-X = #-1.5
"very right"
}

```



## Merging multi-measure rests in a polyphonic part

Multi-measure rests in a polyphonic staff are placed differently depending on the voice they belong to. They can be printed on the same staff line using the setting below. If you omit the `\once` keyword, the change affects all rests in that follow in the given voice.

```
normalPos = \once \revert MultiMeasureRest.direction
```

```

<<
{ c'1 R c'1 \normalPos R c'1 R } \\
{ c'1 R c'1 \normalPos R c'1 R }
>>

```



## Modifying tuplet bracket length

Tuplet brackets can be made to extend horizontally to prefatory matter or the next note. By default, tuplet brackets end at the right edge of the final note of the tuplet; full-length tuplet brackets extend farther to the right, either to cover all the non-rhythmic notation up to the following note, or to cover only the whitespace before the next item of notation, be that a clef, time signature, key signature, or another note. The example shows how to switch tuplets to full length mode and how to modify what material they cover.

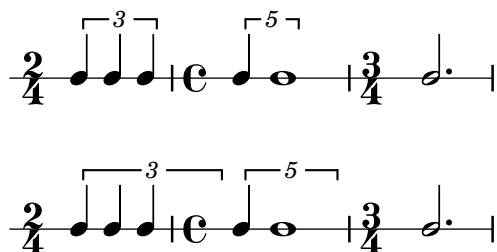
```

\new RhythmicStaff {
  % Defaults.
  \time 2/4 \tuplet 3/2 { c4 4 4 }
  \time 4/4 \tuplet 5/4 { 4 1 }
  \time 3/4 2.
}

\new RhythmicStaff {
  % Set tuplets to be extendable...
  \set tupletFullLength = ##t
  % ...to cover all items up to the next note
  \set tupletFullLengthNote = ##t
  \time 2/4 \tuplet 3/2 { c4 4 4 }
  % ...or to cover just whitespace.
  \set tupletFullLengthNote = ##f
  \time 4/4 \tuplet 5/4 { 4 1 }
  \time 3/4 2.
}

```





## Outputting the version number

It is possible to print the version number of LilyPond in markup.

```
\markup { Processed with LilyPond version #(lilypond-version) }
```

Processed with LilyPond version 2.25.32

## Piano template (simple)

Here is a simple piano staff with some notes.

```
upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  a2 c
}

\score {
  \new PianoStaff \with { instrumentName = "Piano" }
  <<
    \new Staff = "upper" \upper
    \new Staff = "lower" \lower
  >>
  \layout { }
  \midi { }
}
```



## Piano template with centered lyrics

Instead of having a full staff for the melody and lyrics, lyrics can be centered between the staves of a piano staff.

```
upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

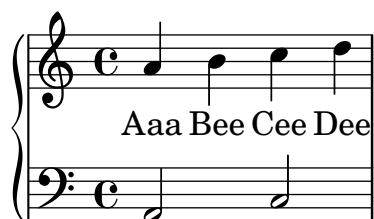
  a4 b c d
}

lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  a2 c
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

\score {
  \new PianoStaff <<
    \new Staff = upper { \new Voice = "singer" \upper }
    \new Lyrics \lyricsto "singer" \text
    \new Staff = lower { \lower }
  >>
  \layout { }
  \midi { }
}
```



## Piano template with melody and lyrics

Here is a typical song format: one staff with the melody and lyrics, with piano accompaniment underneath.

```
melody = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a b c d
}
```

```

text = \lyricmode {
  Aaa Bee Cee Dee
}

upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

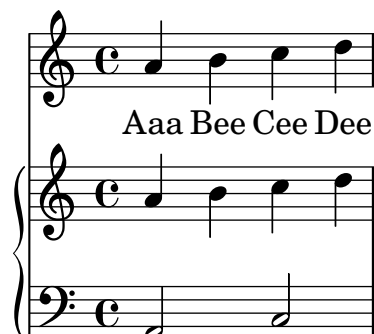
  a4 b c d
}

lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  a2 c
}

\score {
  <<
    \new Voice = "mel" { \autoBeamOff \melody }
    \new Lyrics \lyricsto mel \text
    \new PianoStaff <<
      \new Staff = "upper" \upper
      \new Staff = "lower" \lower
    >>
  >>
  \layout {
    \context { \Staff \RemoveEmptyStaves }
  }
  \midi { }
}

```



## Single-staff template with notes, lyrics, and chords

This template allows the preparation of a song with melody, words, and chords.

```

melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

```

```

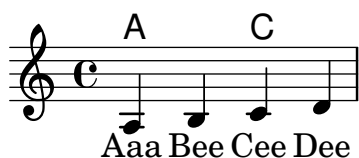
a4 b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

harmonies = \chordmode {
  a2 c
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \harmonies
    }
    \new Voice = "one" { \autoBeamOff \melody }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
  \midi { }
}

```



## Single-staff template with notes and chords

Want to prepare a lead sheet with a melody and chords? Look no further!

```

melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  f4 e8[ c] d4 g |
  a2 ~ a
}

harmonies = \chordmode {
  c4:m f:min7 g:maj c:aug |
  d2:dim b4:5 e:sus
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \harmonies
    }

```

```

    }
    \new Staff \melody
  >>
  \layout{ }
  \midi { }
}

```



## Single-staff template with notes and lyrics

This small template demonstrates a simple melody with lyrics. Cut and paste, add notes, then words for the lyrics. This example turns off automatic beaming, which is common for vocal parts. To use automatic beaming, change or comment out the relevant line.

```

melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

\score{
  <<
    \new Voice = "one" {
      \autoBeamOff
      \melody
    }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
  \midi { }
}

```



## Single-staff template with only notes

This very simple template gives you a staff with notes, suitable for a solo instrument or a melodic fragment. Cut and paste this into a file, add notes, and you're finished!

```

melody = \relative c' {
  \clef treble
  \key c \major

```

```

\time 4/4

a4 b c d
}

\score {
  \new Staff \melody
  \layout { }
  \midi { }
}

```



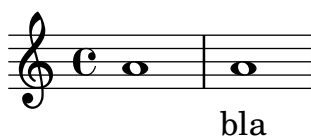
## Skips in lyric mode

The ‘s’ syntax for skips is only available in note mode and chord mode. In other situations, for example, when entering lyrics, using the `\skip` command is recommended.

```

<<
  \relative c'' { a1 | a }
  \new Lyrics \lyricmode { \skip1 bla1 }
>>

```



## Skips in lyric mode (2)

Although ‘s’ skips cannot be used in `\lyricmode` (it is taken to be a literal “s”, not a space), double quotes (“”) or underscores ( \_ ) are available.

```

<<
  \relative c'' { a4 b c d }
  \new Lyrics \lyricmode { a4 "" _ gap }
>>

```



## String quartet template (simple)

This template demonstrates a simple string quartet. It also uses a `\global` section for time and key signatures.

See also snippet “String quartet template with separate parts”.

```

\global= {
  \time 4/4
  \key c \major
}

```

```

violinOne = \new Voice \relative c' {
  c2 d
  e1
  \bar "|."
}

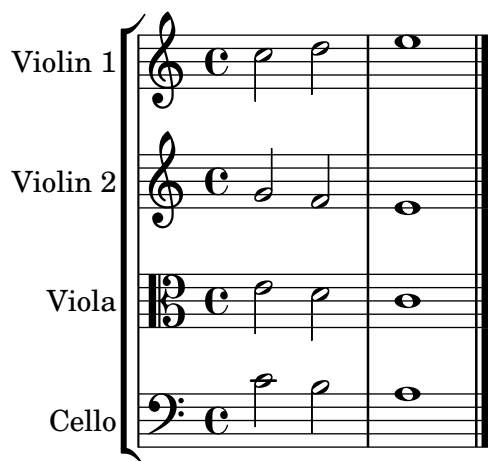
violinTwo = \new Voice \relative c' {
  g2 f
  e1
  \bar "|."
}

viola = \new Voice \relative c' {
  \clef alto
  e2 d
  c1
  \bar "|."
}

cello = \new Voice \relative c' {
  \clef bass
  c2 b
  a1
  \bar "|."
}

\score {
  \new StaffGroup <<
    \new Staff \with { instrumentName = "Violin 1" }
      << \global \violinOne >>
    \new Staff \with { instrumentName = "Violin 2" }
      << \global \violinTwo >>
    \new Staff \with { instrumentName = "Viola" }
      << \global \viola >>
    \new Staff \with { instrumentName = "Cello" }
      << \global \cello >>
  >>
  \layout { }
  \midi { }
}

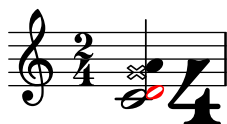
```



## Using the `\tweak` command to tweak individual grobs

With the `\tweak` command, every grob can be tuned directly. Here are some examples of available tweaks.

```
\relative c' {
  \time 2/4
  \set fingeringOrientations = #'(right)
  <
    \tweak font-size #3 c
    \tweak color #red d-\tweak font-size #8 -4
    \tweak style #'cross g
    \tweak duration-log #2 a
  >2
}
```



## Vocal ensemble template

Here is a standard four-part SATB vocal score. With larger ensembles, it is often useful to include a section which is included in all parts. For example, the time signature and key signature are almost always the same for all parts. Like in the “Hymn template”, the four voices are regrouped on only two staves.

```
\paper {
  top-system-spacing.basic-distance = 10
  score-system-spacing.basic-distance = 20
  system-system-spacing.basic-distance = 20
  last-bottom-spacing.basic-distance = 10
}

global = {
  \key c \major
  \time 4/4
}

sopMusic = \relative {
```



```

    c''4 c c8[( b)] c4
}
sopWords = \lyricmode {
    hi hi hi hi
}

altoMusic = \relative {
    e'4 f d e
}
altoWords = \lyricmode {
    ha ha ha ha
}

tenorMusic = \relative {
    g4 a f g
}
tenorWords = \lyricmode {
    hu hu hu hu
}

bassMusic = \relative {
    c4 c g c
}
bassWords = \lyricmode {
    ho ho ho ho
}

\score {
  \new ChoirStaff <<
    \new Lyrics = "sopranos" \with {
      % this is needed for lyrics above a staff
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
    \new Staff = "women" <<
      \new Voice = "sopranos" {
        \voiceOne
        << \global \sopMusic >>
      }
      \new Voice = "altos" {
        \voiceTwo
        << \global \altoMusic >>
      }
    >>
    \new Lyrics = "altos"
    \new Lyrics = "tenors" \with {
      % this is needed for lyrics above a staff
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
    \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" \sopWords
\context Lyrics = "altos" \lyricsto "altos" \altoWords
\context Lyrics = "tenors" \lyricsto "tenors" \tenorWords
\context Lyrics = "basses" \lyricsto "basses" \bassWords
>>
}

```



## Volta brackets in multiple staves

By adding the `Volta_engraver` to the relevant staff, volte can be put over staves other than the topmost one in a score.

`\repeat` and related commands should be present in all staves.

```

voltaMusic = \relative c'' {
  \repeat volta 2 {
    c1
    \alternative {
      \volta 1 { d1 }
      \volta 2 { e1 }
    }
  }
}

<<
\new StaffGroup <<
  \new Staff \voltaMusic
  \new Staff \voltaMusic
>>
\new StaffGroup <<
  \new Staff \with { \consists "Volta_engraver" }
    \voltaMusic
  \new Staff \voltaMusic
>>
>>

```



## 32 Scheme

See also Section “Scheme tutorial” in *Extending* and Section “Interfaces for programmers” in *Extending*.

### Adding extra fingering with Scheme

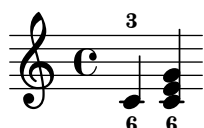
You can add additional elements to notes using `map-some-music`. In this example, an extra script is attached to a note (or a chord).

In general, you should first apply `\displayMusic` to music similar to what you want to create so that you can see its structure. This can be then used as template for your Scheme code.

```
addScript =
#(define-music-function (script music) (ly:event? ly:music?)
  (map-some-music
    (lambda (mus)
      (define (append-script-at! prop)
        (set! (ly:music-property mus prop)
              (append (ly:music-property mus prop)
                      (list (ly:music-deep-copy script)))))
      mus)

    (case (ly:music-property mus 'name)
      ((EventChord)
       (append-script-at! 'elements))
      ((NoteEvent)
       (append-script-at! 'articulations))
      (else #f)))
    music))

{
  \addScript _6 { c'4-3 <c' e' g'> }
}
```



### Adding indicators to staves which get split after a break

This snippet defines the commands `\splitStaffBarLine`, `\convUpStaffBarLine`, and `\convDownStaffBarLine`. These add arrows at a bar line to denote that several voices sharing a staff will each continue on a staff of their own in the next system, or that voices split in this way recombine.

Note that the implementation in this snippet draws dimensionless arrows into the right margin. For normal printing, this doesn’t cause problems. However, it is necessary to increase the bounding box horizontally if you render the code as an image to avoid cropping, as demonstrated below.

```
#(define-markup-command (arrow-at-angle layout props angle-deg length fill)
  (number? number? boolean?)
  (let* ((PI-OVER-180 (/ (atan 1 1) 34))
        (degrees->radians (lambda (degrees) (* degrees PI-OVER-180))))
```

```

        (angle-rad (degrees->radians angle-deg))
        (target-x (* length (cos angle-rad)))
        (target-y (* length (sin angle-rad)))
        (interpret-markup layout props
          (markup
            #:translate (cons (/ target-x 2) (/ target-y 2))
            #:rotate angle-deg
            #:translate (cons (/ length -2) 0)
            #:concat (:#draw-line (cons length 0)
                                  #:arrow-head X RIGHT fill))))))

splitStaffBarLineMarkup = \markup \with-dimensions #'(0 . 0) #'(0 . 0) {
  \combine
  \arrow-at-angle #45 #(sqrt 8) ##t
  \arrow-at-angle #-45 #(sqrt 8) ##t
}

splitStaffBarLine = {
  \once \override Staff.BarLine.stencil =
  #(lambda (grob)
    (ly:stencil-combine-at-edge
      (ly:bar-line::print grob)
      X RIGHT
      (grob-interpret-markup grob splitStaffBarLineMarkup)
      0))
  \break
}

convDownStaffBarLine = {
  \once \override Staff.BarLine.stencil =
  #(lambda (grob)
    (ly:stencil-combine-at-edge
      (ly:bar-line::print grob)
      X RIGHT
      (grob-interpret-markup grob #{
        \markup\with-dimensions #'(0 . 0) #'(0 . 0) {
          \translate #'(0 . -.13)\arrow-at-angle #-45 #(sqrt 8) ##t
        }#}))
      0))
  \break
}

convUpStaffBarLine = {
  \once \override Staff.BarLine.stencil =
  #(lambda (grob)
    (ly:stencil-combine-at-edge
      (ly:bar-line::print grob)
      X RIGHT
      (grob-interpret-markup grob #{
        \markup\with-dimensions #'(0 . 0) #'(0 . 0) {
          \translate #'(0 . .14)\arrow-at-angle #45 #(sqrt 8) ##t
        }#}))
  )
}

```

```

    0))
  \break
}

\paper {
  indent = 10\mm
  short-indent = 10\mm
  line-width = 8\cm
}

separateSopranos = {
  \set Staff.instrumentName = "AI AII"
  \set Staff.shortInstrumentName = "AI AII"
  \splitStaffBarLine
  \change Staff = "up"
}

convSopranos = {
  \convDownStaffBarLine
  \change Staff = "shared"
  \set Staff.instrumentName = "S A"
  \set Staff.shortInstrumentName = "S A"
}

sI = {
  \voiceOne
  \repeat unfold 4 f''2
  \separateSopranos
  \repeat unfold 4 g''2
  \convSopranos
  \repeat unfold 4 c''2
}

sII = {
  s1*2
  \voiceTwo
  \change Staff = "up"
  \repeat unfold 4 d''2
}

aI = {
  \voiceTwo
  \repeat unfold 4 a'2
  \voiceOne
  \repeat unfold 4 b'2
  \convUpStaffBarLine
  \voiceTwo
  \repeat unfold 4 g'2
}

aII = {
  s1*2
  \voiceTwo
  \repeat unfold 4 g'2
}

ten = {

```

```

\voiceOne
\repeat unfold 4 c'2
\repeat unfold 4 d'2
\repeat unfold 4 c'2
}
bas = {
  \voiceTwo
  \repeat unfold 4 f2
  \repeat unfold 4 g2
  \repeat unfold 4 c2
}

\markup \pad-x #3 % avoid cropping
\score {
  <<
    \new ChoirStaff <<
      \new Staff = up \with {
        instrumentName = "SI SII"
        shortInstrumentName = "SI SII"
      } {
        s1*4
      }

      \new Staff = shared \with {
        instrumentName = "S A"
        shortInstrumentName = "S A"
      } <<
        \new Voice = sopI \sI
        \new Voice = sopII \sII
        \new Voice = altI \aI
        \new Voice = altII \aII
      >>
      \new Lyrics \with {
        alignBelowContext = up
      }
      \lyricsto sopII { e f g h }
      \new Lyrics \lyricsto altI { a b c d e f g h i j k l }

      \new Staff = men \with {
        instrumentName = "T B"
        shortInstrumentName = "T B"
      } <<
        \clef F
        \new Voice = ten \ten
        \new Voice = bas \bas
      >>
      \new Lyrics \lyricsto bas { a b c d e f g h i j k l }
    >>
  >>

  \layout {
    \context {

```

```

\Staff \RemoveEmptyStaves
\override VerticalAxisGroup.remove-first = ##t
}
}
}

```

The image displays three musical staves, each with a different number of staves (S A, SI SII, S A) and a different set of notes (a, b, c, d; e, f, g, h; i, j, k, l). The staves are labeled S A, SI SII, and S A. The notes are labeled a, b, c, d; e, f, g, h; i, j, k, l. The staves are grouped by a brace on the left. The first staff (S A) has two staves, the second (SI SII) has three staves, and the third (S A) has two staves. Each staff contains notes with letter labels (a, b, c, d; e, f, g, h; i, j, k, l) and a final double bar line with a repeat sign.

## Adding links to objects

To add a link to a grob stencil you can use `add-link` as defined here. It works both with `\override` and `\tweak`.

Drawback: point-and-click is disturbed for the linked grobs.

Limitation: Works for PDF only.

The linked objects are colored with a separate command.

```

#(define (add-link url-strg)
  (lambda (grob)
    (let* ((stil (ly:grob-property grob 'stencil)))
      (if (ly:stencil? stil)

```



```

    (let* ((x-ext (ly:stencil-extent stil X))
           (y-ext (ly:stencil-extent stil Y))
           (url-expr `(url-link ,url-strg ,x-ext ,y-ext))
           (new-stil
            (ly:stencil-add
             (ly:make-stencil url-expr x-ext y-ext)
             stil)))
      (ly:grob-set-property! grob 'stencil new-stil))))))

%%% test

%% For easier maintenance of this snippet the URL is formatted to use the
%% actually used LilyPond version.
%% Of course a literal URL would work as well.

#(define major.minor-version
  (string-join (take (string-split (lilypond-version) #\.) 2) "."))

urlI =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/writing-pitches"
  major.minor-version)

urlII =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/rhythms"
  major.minor-version)

urlIII =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/note-heads"
  major.minor-version)

urlIV =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/beams"
  major.minor-version)

urlV =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/note-head-styles"
  major.minor-version)

urlVI =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/writing-pitches"
  major.minor-version)

\relative c' {
  \key cis \minor

  \once \override Staff.Clef.color = #green

```

```

\once \override Staff.Clef.after-line-breaking =
  #(add-link urlI)

\once \override Staff.TimeSignature.color = #green
\once \override Staff.TimeSignature.after-line-breaking =
  #(add-link urlII)

\once \override NoteHead.color = #green
\once \override NoteHead.after-line-breaking =
  #(add-link urlIII)

cis'1
\once \override Beam.color = #green
\once \override Beam.after-line-breaking =
  #(add-link urlIV)
cis8 dis e fis gis2
<gis,
  \tweak Accidental.color #green
  \tweak Accidental.after-line-breaking #(add-link urlVI)
  \tweak color #green
  \tweak after-line-breaking #(add-link urlV)
  \tweak style #'harmonic
bis
dis
fis
>1
<cis, cis' e>
}

```



## Adding orchestral cues to a vocal score

This snippet shows one approach to simplify adding many orchestral cues to the piano reduction in a vocal score. The music function `\cueWhile` takes four arguments: the music from which the cue is to be taken, as defined by `\addQuote`, the name to be inserted before the cue notes, then either UP or DOWN to specify either `\voiceOne` with the name above the staff or `\voiceTwo` with the name below the staff, and finally the piano music in parallel with which the cue notes are to appear. The name of the cued instrument is positioned to the left of the cued notes. Many passages can be cued, but they cannot overlap each other in time.

```

cueWhile =
#(define-music-function
  (instrument name dir music)
  (string? string? ly:dir? ly:music?)
  #{
    \cueDuring $instrument #dir {
      \once \override TextScript.self-alignment-X = #RIGHT
      \once \override TextScript.direction = $dir
      <>-\markup { \tiny #name }
      $music
    }
  })

```

```

    }
    #})

flute = \relative c'' {
  \transposition c'
  s4 s4 e g
}
\addQuote "flute" { \flute }

clarinet = \relative c' {
  \transposition bes
  fis4 d d c
}
\addQuote "clarinet" { \clarinet }

singer = \relative c'' { c4. g8 g4 bes4 }
words = \lyricmode { here's the lyr -- ics }

pianoRH = \relative c'' {
  \transposition c'
  \cueWhile "clarinet" "Clar." #DOWN { c4. g8 }
  \cueWhile "flute" "Flute" #UP { g4 bes4 }
}
pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  <<
    \new Staff {
      \new Voice = "singer" {
        \singer
      }
    }
    \new Lyrics {
      \lyricsto "singer"
      \words
    }
    \new PianoStaff <<
      \new Staff {
        \new Voice {
          \pianoRH
        }
      }
      \new Staff {
        \clef "bass"
        \pianoLH
      }
    >>
  >>
}

```



## Adding the current date to a score

With a little Scheme code, the current date can easily be added to a score.

```
\paper { tagline = ##f }
```

```
% first, define a variable to hold the formatted date:
```

```
date = #(\strptime "%d-%m-%Y" (localtime (current-time)))
```

```
% use it in the title block:
```

```
\header {
```

```
  title = "Including the date!"
```

```
  subtitle = \date
```

```
}
```

```
\score {
```

```
  \relative c' {
```

```
    c4 c c c
```

```
  }
```

```
}
```

```
% and use it in a \markup block:
```

```
\markup {
```

```
  \date
```

```
}
```

## Including the date!

**17-01-2026**



17-01-2026

## Adjusting slur positions vertically

Using `\override Slur.positions` it is possible to set the vertical position of the start and end points of a slur to absolute values (or rather, forcing LilyPond's slur algorithm to consider these values as desired). In many cases, this means a lot of trial and error until good values are found. You probably have tried the `\offset` command next just to find out that it doesn't work for slurs, emitting a warning instead.

The code in this snippet allows you to tweak the vertical start and end positions by specifying *relative* changes, similar to `\offset`.

```

Syntax: \offsetPositions #'(dy1 . dy2)

offsetPositions =
#(define-music-function (offsets) (number-pair?)
  #{
    \once \override Slur.control-points =
      #(lambda (grob)
        (match-let (((_ . y1) _ _ (_ . y2))
                    (ly:slur::calc-control-points grob))
          ((off1 . off2) offsets))
        (set! (ly:grob-property grob 'positions)
              (cons (+ y1 off1) (+ y2 off2)))
        (ly:slur::calc-control-points grob)))
      #})

\relative c' ' {
  c4(^"default" c, d2)
  \offsetPositions #'(0 . 1)
  c'4(^"(0 . 1)" c, d2)
  \offsetPositions #'(0 . 2)
  c'4(^"(0 . 2)" c, d2)
  \bar "||"
  g4(^"default" a d'2)
  \offsetPositions #'(1 . 0)
  g,,4(^"(1 . 0)" a d'2)
  \offsetPositions #'(2 . 0)
  g,,4(^"(2 . 0)" a d'2)
}

```



## Center text below hairpin dynamics

This example provides a function to typeset a hairpin (de)crescendo with some additional text below it, such as “molto” or “poco”. The added text will change the direction according to the direction of the hairpin. The Hairpin is aligned to a DynamicText grob.

The example also illustrates how to modify the way an object is normally printed, using some Scheme code.

```

hairpinWithCenteredText =
#(define-music-function (text) (markup?)
  #{
    \once \override Voice.Hairpin.after-line-breaking =
      #(lambda (grob)
        (let* ((stencil (ly:hairpin::print grob))
              (par-y (ly:grob-parent grob Y))
              (dir (ly:grob-property par-y 'direction))
              (staff-line-thickness
                (ly:output-def-lookup (ly:grob-layout grob)
                                      'line-thickness)))
          stencil
          (par-y (ly:grob-parent grob Y))
          (dir (ly:grob-property par-y 'direction))
          (staff-line-thickness
            (ly:output-def-lookup (ly:grob-layout grob)
                                  'line-thickness)))
        (ly:hairpin::print grob))
      #})

```

```

(new-stencil
  (ly:stencil-aligned-to
    (ly:stencil-combine-at-edge
      (ly:stencil-aligned-to stencil X CENTER)
      Y dir
      (ly:stencil-aligned-to
        (grob-interpret-markup
          grob
          (make-fontsize-markup
            (magnification->font-size
              (+ (ly:staff-symbol-staff-space grob)
                (/ staff-line-thickness 2)))
            text))
        X CENTER))
      X LEFT))
  (staff-space (ly:output-def-lookup
    (ly:grob-layout grob) 'staff-space))
  (par-x (ly:grob-parent grob X))
  (dyn-text (grob::has-interface par-x
    'dynamic-text-interface))

  (dyn-text-stencil-x-length
    (if dyn-text
      (interval-length
        (ly:stencil-extent
          (ly:grob-property par-x 'stencil) X))
      0))
  (x-shift
    (if dyn-text (- (+ staff-space dyn-text-stencil-x-length)
      (* 0.5 staff-line-thickness))
    0)))

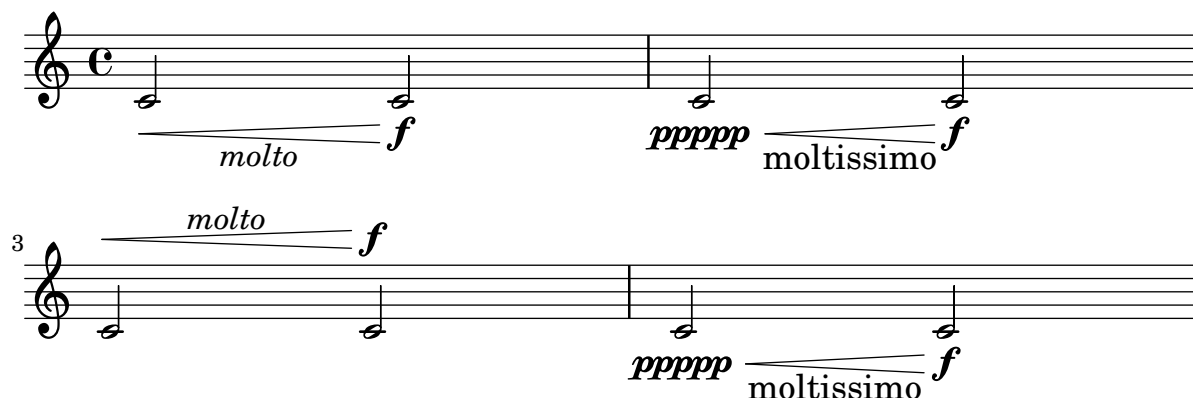
(ly:grob-set-property! grob 'Y-offset 0)
(ly:grob-set-property! grob
  'stencil (ly:stencil-translate-axis
    new-stencil
    x-shift X))))

#})

hairpinMolto = \hairpinWithCenteredText \markup { \italic molto }
hairpinMore = \hairpinWithCenteredText \markup { \larger moltissimo }

\relative c' {
  \hairpinMolto c2\< c\f
  \hairpinMore c2\ppppp\< c\f
  \break
  \hairpinMolto c2^\< c\f
  \hairpinMore c2\ppppp\< c\f
}

```



## Changing properties for individual grobs

The `\applyOutput` command allows the tuning of any layout object, in any context. It requires a Scheme function with three arguments.

In the example below, function `mc-squared` is executed for all `NoteHead` grobs (within the current `Voice` context) at the current time step; the function modifies the grob's `stencil`, using the `staff-position` property to replace some pitches with markup.

See the 'Extending' manual (<https://lilypond.org/doc/v2.24/Documentation/extending/running-a-function-on-all-layout-objects>) for more information.

```
#(define (mc-squared grob grob-origin context)
  (let ((sp (ly:grob-property grob 'staff-position)))
    (ly:grob-set-property!
      grob 'stencil
      (grob-interpret-markup grob
        #{ \markup \lower #0.5
          #(case sp
              ((-5) "m")
              ((-3) "c ")
              ((-2) #{ \markup \teeny \bold 2 #})
              (else "bla")) #}))))
```

```
\relative c' {
  <d f g b>2
  \applyOutput Voice.NoteHead #mc-squared
  <d f g b>2
}
```



## Chord name exceptions

The property `chordNameExceptions` stores a list of chord name exceptions to handle cases either not covered or handled incorrectly.

The default chord names used by LilyPond follow the rules as given in Klaus Ignatzek's book "Die Jazzmethode für Klavier 1"; the algorithm to convert chords to chord names can be found in file `scm/chord-ignatzek-names.scm`. Additional rules are given as chord exceptions and stored in the variable `ignatzekExceptions`, as set up in file `ly/chord-modifiers-init.ly`.

This snippet modifies these exceptions in three steps.

1. Set up some music with chords and associated markup. By convention, the root (i.e., the lowest note) of each chord should have pitch c.
2. Call Scheme function `sequential-music-to-chord-exceptions` to create a new list of exceptions, then concatenate it with the existing ones. Since `ignatzekExceptions` is set up with this function's second parameter set to `#t` (to ignore the root of the chords), we have to do the same.
3. Register the new exception list.

*% Step 1: Define music with chords and markup for maj9 and 6(add9).*

```
chExceptionMusic = {
  <c e g b d'>-\markup { \super "maj9" }
  <c e g a d'>-\markup { \super "6(add9)" }
}
```

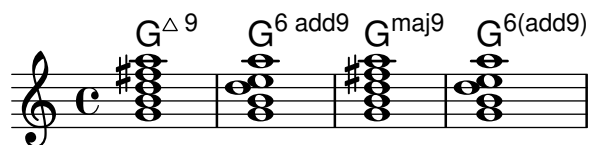
*% Step 2: Create extended exception list.*

```
chExceptions =
#(append (sequential-music-to-chord-exceptions chExceptionMusic #t)
         ignatzekExceptions)
```

```
theMusic = \chordmode {
  g1:maj9 g1:6.9
  % Step 3: Register extended exception list.
  \set chordNameExceptions = #chExceptions
  g1:maj9 g1:6.9
}
```

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

```
<<
  \new ChordNames \theMusic
  \new Voice \theMusic
>>
```



## Coloring notes depending on their pitch

It is possible to color note heads depending on their pitch and/or their names: the function used in this example even makes it possible to distinguish enharmonics.

*% Association list of pitches to colors.*

```
#(define color-mapping
  (list
    (cons (ly:make-pitch 0 0 NATURAL) (x11-color 'red))
    (cons (ly:make-pitch 0 0 SHARP) (x11-color 'green))
    (cons (ly:make-pitch 0 1 FLAT) (x11-color 'green))
    (cons (ly:make-pitch 0 2 NATURAL) (x11-color 'red))
```



```

(cons (ly:make-pitch 0 2 SHARP) (x11-color 'green))
(cons (ly:make-pitch 0 3 FLAT) (x11-color 'red))
(cons (ly:make-pitch 0 3 NATURAL) (x11-color 'green))
(cons (ly:make-pitch 0 4 SHARP) (x11-color 'red))
(cons (ly:make-pitch 0 5 NATURAL) (x11-color 'green))
(cons (ly:make-pitch 0 5 FLAT) (x11-color 'red))
(cons (ly:make-pitch 0 6 SHARP) (x11-color 'red))
(cons (ly:make-pitch 0 1 NATURAL) (x11-color 'blue))
(cons (ly:make-pitch 0 3 SHARP) (x11-color 'blue))
(cons (ly:make-pitch 0 4 FLAT) (x11-color 'blue))
(cons (ly:make-pitch 0 5 SHARP) (x11-color 'blue))
(cons (ly:make-pitch 0 6 FLAT) (x11-color 'blue))))

% Compare pitch and alteration (not octave).
#(define (pitch-equals? p1 p2)
  (and
    (= (ly:pitch-alteration p1) (ly:pitch-alteration p2))
    (= (ly:pitch-notename p1) (ly:pitch-notename p2))))

#(define (pitch-to-color pitch)
  (let ((color (assoc pitch color-mapping pitch-equals?)))
    (if color
      (cdr color))))

#(define (color-notehead grob)
  (pitch-to-color
    (ly:event-property (event-cause grob) 'pitch)))

\score {
  \new Staff \relative c' {
    \override NoteHead.color = #color-notehead
    c8 b d dis ees f g aes
  }
}

```



## Creating “real” parenthesized dynamics

Although the easiest way to add parentheses to a dynamic mark is to use a `\markup` block, this method has a downside: the created objects behave like text markups and not like dynamics.

However, it is possible to create a similar object using the equivalent Scheme code (as described in the Notation Reference), combined with the `make-dynamic-script` function. This way, the markup is regarded as a dynamic and therefore remains compatible with commands such as `\dynamicUp` or `\dynamicDown`.

```

paren =
#(define-event-function (dyn) (ly:event?)
  (make-dynamic-script
    #{ \markup \concat {
      \normal-text \italic \fontsize #2 (

```

```

\pad-x #0.2 #(ly:music-property dyn 'text)
\normal-text \italic \fontsize #2 )
}
#})))

\relative c'' {
  c4\paren\f c c \dynamicUp c\paren\p
}

```



## Creating a sequence of notes on various pitches

In music that contains many occurrences of the same sequence of notes at different pitches, the following music function may prove useful. It takes a note, of which only the pitch is used.

This example creates the rhythm used throughout *Mars*, from Gustav Holst's *The Planets*.

```

rhythm =
#(define-music-function (p) (ly:pitch?)
  "Make the rhythm in Mars (the Planets) at the given pitch"
  #{ \tuplet 3/2 { $p 8 8 8 } 4 4 8 8 4 #})

\new Staff {
  \time 5/4
  \rhythm c'
  \rhythm c''
  \rhythm g
}

```



## Creating custom dynamics in MIDI output

The following example shows how to create a dynamic marking, not included in the default list, and assign a specific value to it so that it affects MIDI output.

The dynamic mark `\rfz` gets value 0.9.

```

#(define (myDynamics dynamic)
  (if (equal? dynamic "rfz")
    0.9
    (default-dynamic-absolute-volume dynamic)))

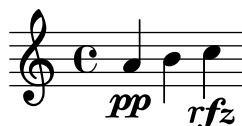
\score {
  \new Staff {
    \set Staff.midiInstrument = "cello"
    \set Score.dynamicAbsoluteVolumeFunction = #myDynamics
    \new Voice {
      \relative {

```

```

        a'4\pp b c-\rfz
      }
    }
  }
  \layout {}
  \midi {}
}

```



## Customizing the position and number of dots in repeat sign bar lines

If you want to customize the position and/or number of dots in repeat sign bar lines, you can define new custom bar lines or redefine the way default repeat signs are drawn. This snippet shows how. This may be particularly helpful when using a staff with custom line-positions, as shown in this snippet.

```

#(define ((make-custom-dot-bar-line dot-positions) is-span grob extent)

  "Draw dots (repeat sign dots) at @var{dot-positions}. The
  coordinates of @var{dot-positions} are equivalent to the
  coordinates of @code{StaffSymbol.line-positions}, a dot-position
  of X and a line-position of X indicate the same vertical position."

  (let* ((staff-space (ly:staff-symbol-staff-space grob))
        (dot (ly:font-get-glyph (ly:grob-default-font grob) "dots.dot"))
        (stencil empty-stencil))
    (for-each
      (lambda (dp)
        (set! stencil (ly:stencil-add stencil
                                      (ly:stencil-translate-axis dot (* dp (/ staff-space 2)) Y))))
      dot-positions)
    stencil))

% With the procedure above we can define custom bar-lines, for example,
% one that resembles standard repeat sign bar lines except it has
% three dots at staff positions -3, 0, and 3.

#(add-bar-glyph-print-procedure "*" (make-custom-dot-bar-line '(-3 0 3)))
\defineBarLine ".|*" #'(" " "*" " ")
\defineBarLine "*|. " #'(" " "*" " ")

% We can also customize the dot positions used in all default repeat signs
% by redefining the print procedure of the colon bar glyph (":"). On a staff
% with line-positions of '(-4 -2 2 4) the default repeat sign dots appear
% at '(-3 3), but we can put them at '(-1 1) instead.

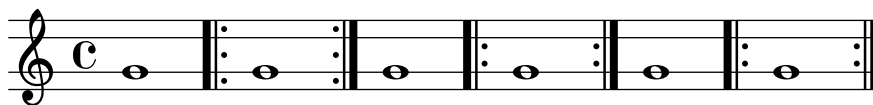
#(add-bar-glyph-print-procedure ":" (make-custom-dot-bar-line '(-1 1)))

```

```

\new Staff \with {
  \override StaffSymbol.line-positions = #'(-4 -2 2 4)
  \override StaffSymbol.staff-space = #1.3
} {
  \relative f' {
    g1
    \bar ".*|"
    g
    \bar "*|."
    g
    \bar ".|:-|"
    g
    \bar ":|."
    g
    \repeat volta 2 {
      g
    }
  }
}

```



## Defining an engraver in Scheme: ambitus engraver

This example demonstrates how the ambitus engraver may be defined on the user side, with a Scheme engraver. This is basically a rewrite in Scheme of the code from `lily/ambitus-engraver.cc`.

```

#(use-modules (oop goops))

```

```

%%%
%%% Grob utilities
%%%
%%% These are literal rewrites of some C++ methods used by the ambitus
%%% engraver.

```

```

#(define (ly:separation-item::add-conditional-item grob grob-item)
  "Add GROB-ITEM to the array of conditional elements of GROB.

```

This is a rewrite of function ``Separation_item::add_conditional_item`` from file ``lily/separation-item.cc``."

```

  (ly:pointer-group-interface::add-grob
    grob 'conditional-elements grob-item))

```

```

#(define (ly:accidental-placement::accidental-pitch accidental-grob)
  "Get the pitch from the grob cause of ACCIDENTAL-GROB.

```

This is a rewrite of function ``accidental_pitch`` from file ``lily/accidental-placement.cc``."

```

  (ly:event-property (ly:grob-property

```

```

        (ly:grob-parent accidental-grob Y) 'cause)
        'pitch))

#(define (ly:accidental-placement::add-accidental grob accidental-grob)
  "Add ACCIDENTAL-GROB to the list of accidentals grobs of GROB.
  ACCIDENTAL-GROB is an `Accidental` grob; GROB is an `AccidentalPlacement`
  grob.

  This is a rewrite of function `Accidental_placement::add_accidental` from
  file `lily/accidental-placement.cc`."
  (let ((pitch (ly:accidental-placement::accidental-pitch
    accidental-grob)))
    (set! (ly:grob-parent accidental-grob X) grob)
    (let* ((accidentals (ly:grob-object grob 'accidental-grobs))
      (handle (assq (ly:pitch-notename pitch) accidentals))
      (entry (if handle (cdr handle) '())))
      (set! (ly:grob-object grob 'accidental-grobs)
        (assq-set! accidentals
          (ly:pitch-notename pitch)
          (cons accidental-grob entry))))))

%%%
%%% Ambitus data structures.
%%%

%%% The <ambitus> class holds the various grobs that are created to
%%% print an ambitus:
%%%
%%% - `ambitus-group`: the grob that groups all the components of an
%%%   ambitus (`Ambitus` grob);
%%% - `ambitus-line`: the vertical line between the upper and lower
%%%   ambitus notes (`AmbitusLine` grob);
%%% - `ambitus-up-note` and `ambitus-down-note`: the note head and
%%%   accidental for the lower and upper note of the ambitus (see
%%%   `` class below).
%%%
%%% The other slots define the key and clef context of the engraver:
%%%
%%% - `start-c0`: position of middle c at the beginning of the piece.
%%%   It is used to place the ambitus notes according to their pitch;
%%% - `start-key-sig`: the key signature at the beginning of the
%%%   piece. It is used to determine whether accidentals shall be
%%%   printed next to ambitus notes.

#(define-class <ambitus> ()
  (ambitus-group #:accessor ambitus-group)
  (ambitus-line #:accessor ambitus-line)
  (ambitus-up-note #:getter ambitus-up-note
    #:init-form (make <ambitus-note>))
  (ambitus-down-note #:getter ambitus-down-note
    #:init-form (make <ambitus-note>))
  (start-c0 #:accessor ambitus-start-c0)

```

```

      #:init-value #f)
    (start-key-sig #:accessor ambitus-start-key-sig
      #:init-value '()))

%%% Accessor for the lower and upper note data of an ambitus.
#(define-method (ambitus-note (ambitus <ambitus>) direction)
  "Return lower or upper note of AMBITUS depending on DIRECTION."
  (if (= direction UP)
      (ambitus-up-note ambitus)
      (ambitus-down-note ambitus)))

%%% The `ambitus-note` class holds the grobs that are specific to
%%% ambitus (lower and upper) notes:
%%%
%%% - `head`: an `AmbitusNoteHead` grob;
%%% - `accidental`: an `AmbitusAccidental` grob, to be possibly
%%%   printed next to the ambitus note head.
%%%
%%% Moreover,
%%%
%%% - `pitch` is the absolute pitch of the note;
%%% - `cause` is the note event that causes this ambitus note, i.e.,
%%%   the lower or upper note of the considered music sequence.

#(define-class <ambitus-note> ()
  (head #:accessor ambitus-note-head
    #:init-value #f)
  (accidental #:accessor ambitus-note-accidental
    #:init-value #f)
  (cause #:accessor ambitus-note-cause
    #:init-value #f)
  (pitch #:accessor ambitus-note-pitch
    #:init-value #f))

%%%
%%% Ambitus engraving logic.
%%%
%%% This is rewrite of the code from file `lily/ambitus-engraver.cc`.

#(define (make-ambitus translator)
  "Build an ambitus object: initialize all the grobs and their
relations."

```

The `Ambitus` grob contains all other grobs:

```

Ambitus
|- AmbitusLine
|- AmbitusNoteHead    for upper note
|- AmbitusAccidental  for upper note
|- AmbitusNoteHead    for lower note
|- AmbitusAccidental  for lower note

```

The parent of an accidental is the corresponding note head, and the accidental is set as the ``accidental-grob`` property of the note head so that is printed by the function that prints notes."

```
;; Make the ambitus object.
(let ((ambitus (make <ambitus>)))
  ;; Build the `Ambitus` grob, which will contain all other grobs.
  (set! (ambitus-group ambitus)
        (ly:engraver-make-grob translator 'Ambitus '()))
  ;; Build the `AmbitusLine` grob (the line between lower and upper
  ;; note).
  (set! (ambitus-line ambitus)
        (ly:engraver-make-grob translator 'AmbitusLine '()))
  ;; Build the upper and lower `AmbitusNoteHead` and
  ;; `AmbitusAccidental`.
  (for-each
   (lambda (direction)
     (let ((head (ly:engraver-make-grob translator
                                             'AmbitusNoteHead '()))
           (accidental (ly:engraver-make-grob translator
                                                  'AmbitusAccidental '()))
           (group (ambitus-group ambitus)))
       ;; The parent of the `AmbitusAccidental` grob is the
       ;; `AmbitusNoteHead` grob.
       (set! (ly:grob-parent accidental Y) head)
       ;; The `AmbitusAccidental` grob is set as the
       ;; `accidental-grob` object of `AmbitusNoteHead`. This is
       ;; later used by the function that prints notes.
       (set! (ly:grob-object head 'accidental-grob) accidental)
       ;; Both the note head and the accidental grobs are added to
       ;; the main ambitus grob.
       (ly:axis-group-interface::add-element group head)
       (ly:axis-group-interface::add-element group accidental)
       ;; The note head and the accidental grobs are added to the
       ;; ambitus object.
       (set! (ambitus-note-head (ambitus-note ambitus direction))
             head)
       (set! (ambitus-note-accidental (ambitus-note ambitus direction))
             accidental)))
     (list DOWN UP))

   ;; The parent of the ambitus line is the lower ambitus note head.
   (set! (ly:grob-parent (ambitus-line ambitus) X)
         (ambitus-note-head (ambitus-note ambitus DOWN)))
   ;; The ambitus line is added to the ambitus main grob.
   (ly:axis-group-interface::add-element (ambitus-group ambitus)
                                           (ambitus-line ambitus))

  ambitus))
```

```
#(define-method (initialize-ambitus-state
```

```
  (ambitus <ambitus>) translator)
```

```
"Initialize the state of AMBITUS by getting the starting position of
middle C and key signature from TRANSLATOR's context."
```

```

(if (not (ambitus-start-c0 ambitus))
  (begin
    (set! (ambitus-start-c0 ambitus)
      (ly:context-property (ly:translator-context translator)
        'middleCPosition 0))
    (set! (ambitus-start-key-sig ambitus)
      (ly:context-property (ly:translator-context translator)
        'keyAlterations))))))

#(define-method (update-ambitus-notes (ambitus <ambitus>) note-grob)
  "Update upper and lower ambitus pitches of AMBITUS using NOTE-GROB."
  ;; Get the event that caused the `note-grob` creation and check
  ;; that it is a `note-event`.
  (let ((note-event (ly:grob-property note-grob 'cause)))
    (if (ly:in-event-class? note-event 'note-event)
      ;; Get the pitch from the note event.
      (let ((pitch (ly:event-property note-event 'pitch)))
        ;; If this pitch is lower than the current ambitus' lower
        ;; note pitch (or it has not been initialized yet), then
        ;; this pitch is the new ambitus' lower pitch. The same is
        ;; done for the upper pitch (but in the opposite
        ;; direction).
        (for-each
          (lambda (direction pitch-compare)
            (if (or (not (ambitus-note-pitch
              (ambitus-note ambitus direction)))
              (pitch-compare
                pitch (ambitus-note-pitch
                  (ambitus-note ambitus direction)))))
              (begin
                (set! (ambitus-note-pitch
                  (ambitus-note ambitus direction))
                  pitch)
                (set! (ambitus-note-cause
                  (ambitus-note ambitus direction))
                  note-event))))
            (list DOWN UP)
            (list ly:pitch<?
              (lambda (p1 p2) (ly:pitch<? p2 p1)))))))
        (list ly:pitch<?
          (lambda (p1 p2) (ly:pitch<? p2 p1)))))))

#(define-method (typeset-ambitus (ambitus <ambitus>) translator)
  "Typeset AMBITUS.

```

- Place the lower and upper ambitus notes according to their pitch and the position of the middle C.
- Typeset or delete the note accidentals, according to the key signature. An accidental, if it is to be printed, is added to an `AccidentalPlacement` grob (a grob dedicated to the placement of accidentals near a chord).
- Both note heads are added to the ambitus line grob so that a line gets printed between them."
  - ;; Check whether there are lower and upper pitches.



```

(if (and (ambitus-note-pitch (ambitus-note ambitus UP))
        (ambitus-note-pitch (ambitus-note ambitus DOWN)))
    ;; Make an `AccidentalPlacement` grob, for placement of note
    ;; accidentals.
    (let ((accidental-placement
          (ly:engraver-make-grob
           translator
           'AccidentalPlacement (ambitus-note-accidental
                                 (ambitus-note ambitus DOWN)))))
      ;; For lower and upper ambitus notes.
      (for-each
       (lambda (direction)
         (let ((pitch (ambitus-note-pitch
                        (ambitus-note ambitus direction))))
           ;; Set the cause and the staff position of the ambitus
           ;; note according to the associated pitch.
           (set! (ly:grob-property
                  (ambitus-note-head (ambitus-note ambitus direction))
                  'cause)
                 (ambitus-note-cause (ambitus-note ambitus direction)))
           (set! (ly:grob-property
                  (ambitus-note-head (ambitus-note ambitus direction))
                  'staff-position)
                 (+ (ambitus-start-c0 ambitus)
                    (ly:pitch-steps pitch)))
           ;; Determine whether an accidental shall be printed for
           ;; this note, according to the key signature.
           (let* ((handle
                   (or (assoc (cons (ly:pitch-octave pitch)
                                   (ly:pitch-notename pitch))
                           (ambitus-start-key-sig ambitus))
                       (assoc (ly:pitch-notename pitch)
                              (ambitus-start-key-sig ambitus))))
                  (sig-alter (if handle (cdr handle) 0)))
             (cond
              ((= (ly:pitch-alteration pitch) sig-alter)
               ;; The note alteration is in the key signature
               ;; => it does not have to be printed.
               (ly:grob-suicide! (ambitus-note-accidental
                                 (ambitus-note ambitus direction)))
               (set! (ly:grob-object (ambitus-note-head
                                     (ambitus-note ambitus direction))
                                 'accidental-grob)
                     '()))
              (else
               ;; Otherwise the accidental shall be printed.
               (set! (ly:grob-property
                      (ambitus-note-accidental
                       (ambitus-note ambitus direction)) 'alteration)
                     (ly:pitch-alteration pitch))))))
           ;; Add the `AccidentalPlacement` grob to the conditional
           ;; items of the `AmbitusNoteHead`.

```

```

    (ly:separation-item::add-conditional-item
      (ambitus-note-head (ambitus-note ambitus direction))
      accidental-placement)
    ;; Add the `AmbitusAccidental` to the list of the
    ;; `AccidentalPlacement` grob accidentals.
    (ly:accidental-placement::add-accidental
      accidental-placement
      (ambitus-note-accidental (ambitus-note ambitus direction)))
    ;; Add the `AmbitusNoteHead` grob to the `AmbitusLine` grob.
    (ly:pointer-group-interface::add-grob
      (ambitus-line ambitus)
      'note-heads
      (ambitus-note-head (ambitus-note ambitus direction))))
  (list DOWN UP))
;; Add the `AccidentalPlacement` grob to the main `Ambitus` grob.
(ly:axis-group-interface::add-element
  (ambitus-group ambitus) accidental-placement))
;; No lower and upper pitches => nothing to print.
(begin
  (for-each
    (lambda (direction)
      (ly:grob-suicide! (ambitus-note-accidental
        (ambitus-note ambitus direction)))
      (ly:grob-suicide! (ambitus-note-head
        (ambitus-note ambitus direction)))))
    (list DOWN UP))
  (ly:grob-suicide! ambitus-line))))

%%%
%%% Ambitus engraver definition.
%%%
#(define ambitus-engraver
  (lambda (context)
    (let ((ambitus #f))
      ;; When music is processed, make the ambitus object if not
      ;; already built.
      (make-engraver
        ((process-music translator)
          (if (not ambitus)
              (set! ambitus (make-ambitus translator))))

        ;; Set the ambitus clef and key signature state.
        ((stop-translation-timestep translator)
          (if ambitus
              (initialize-ambitus-state ambitus translator)))

        ;; When a note head grob is built, update the ambitus notes.
        (acknowledgers
          ((note-head-interface engraver grob source-engraver)
            (if ambitus
              (update-ambitus-notes ambitus grob))))))

```

```

;; Finally, typeset the ambitus according to its upper and
;; lower notes (if any).
((finalize translator)
  (if ambitus
    (typeset-ambitus ambitus translator))))))

%%%
%%% Example
%%%

\score {
  \new StaffGroup <<
    \new Staff { c'4 des' e' fis' gis' }
    \new Staff { \clef "bass" c4 des ~ des ees b, }
  >>
  \layout { \context { \Staff \consists #ambitus-engraver } }
}

```



## Different font size settings for instrumentName and shortInstrumentName

Choose different font sizes for `instrumentName` and `shortInstrumentName` as a context override.

```
InstrumentNameFontSize =
#(define-music-function (font-size-pair) (pair?)
  "Set the font size of `InstrumentName` grobs.
```

The first value of FONT-SIZE-PAIR sets the font size of the initial ``instrumentName`` property, the second value sets the font size of ``shortInstrumentName``."

[illegible]

```

                                (cdr siblings))))))
#})

\layout {
  indent = 3\cm
  short-indent = 0.8\cm

  \context {
    \Staff
    \InstrumentNameFontSize #'(6 . -3)
  }
}

\new StaffGroup <<
  \new Staff \with {
    instrumentName = "Flute"
    shortInstrumentName = "Fl." } {
    c''1 \break c'' \break c'' }
  \new Staff \with {
    instrumentName = "Violin"
    shortInstrumentName = "Vl." } {
    c''1 \break c'' \break c'' }
>>

```

Flute

Violin

Fl.

VI.

Fl.

VI.

## Displaying grob ancestry

When working with grob callbacks, it can be helpful to understand a grob's ancestry. Most grobs have parents which influence the positioning of the grob. X- and Y-parents influence the horizontal and vertical positions for the grob, respectively. Additionally, each parent may have parents of its own.

- The types of parents a grob has may depend on context.
- For some grobs, the X- and Y-parents are the same.
- A particular *ancestor* may be related to a grob in multiple ways.
- The concept of *generations* is misleading.

The macro defined in this snippet prints (to the console) a textual representation of a grob's ancestry. For example, the call

generates the following output.

As a consequence, you have to execute the code in this snippet by yourself, since the generated output file doesn't show the data we are interested in.

```
#(define (get-ancestry grob)
  (if (not (null? (ly:grob-parent grob X)))
      (list (grob::name grob)
            (get-ancestry (ly:grob-parent grob X))
            (get-ancestry (ly:grob-parent grob Y)))
      (grob::name grob)))

#(define (format-ancestry lst padding)
  (string-append
    (symbol->string (car lst)) "\n"
    (let ((X-ancestry (if (list? (cadr lst))
                          (format-ancestry (cadr lst) (+ padding 3))
                          (symbol->string (cadr lst))))
          (Y-ancestry (if (list? (caddr lst))
                          (format-ancestry (caddr lst) (+ padding 3))
                          (symbol->string (caddr lst)))))
      (if (equal? X-ancestry Y-ancestry)
          (string-append (format #f "~&")
                        (make-string padding #\space)
                        "X,Y: "
                        (if (list? (cadr lst))
```

```

                (format-ancestry (cadr lst) (+ padding 5))
                (symbol->string (cadr lst))))
  (string-append (format #f "~&")
    (make-string padding #\space)
    "X: " X-ancestry "\n"
    (make-string padding #\space)
    "Y: " Y-ancestry (format #f "~&"))))
  (format #f "~&"))))

#(define (display-ancestry grob)
  (format (current-output-port)
    "~2&~a~2%~a~&"
    (make-string 36 #\-)
    (if (ly:grob? grob)
      (format-ancestry (get-ancestry grob) 0)
      (format #f "~a is not a grob" grob))))

\relative c' {
  \once \override NoteHead.before-line-breaking = #display-ancestry
  f4
  \once \override Accidental.before-line-breaking = #display-ancestry
  \once \override Arpeggio.before-line-breaking = #display-ancestry
  <f as c>4\arpeggio
}

```



## Drawing circles around note heads

A circle can be drawn around a note head by providing a custom Scheme function to temporarily override the stencil property.

```

circle = \tweak NoteHead.stencil
  #(lambda (grob)
    (let* ((note (ly:note-head::print grob))
      (combo-stencil (ly:stencil-add
        note
        (circle-stencil note 0.1 0.8))))
      (ly:make-stencil (ly:stencil-expr combo-stencil)
        (ly:stencil-extent note X)
        (ly:stencil-extent note Y))))
  \etc

{ \circle c' }

```



## Drawing circles around various objects

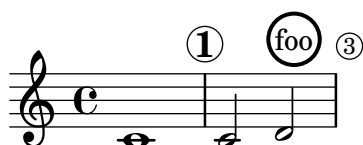
The `\circle` command draws circles around `\markup` objects. For other objects, specific tweaks may be required, as demonstrated for rehearsal marks and measure numbers.

```

\relative c' {
  c1
  \set Score.rehearsalMarkFormatter =
    #(lambda (mark context)
      (make-circle-markup (format-mark-numbers mark context)))
  \mark \default

  c2 d^\markup {
    \override #'(thickness . 3) {
      \circle foo
    }
  }
  \override Score.BarNumber.break-visibility = #all-visible
  \override Score.BarNumber.stencil =
    #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
}

```



## Extending glissandi across repeats

A glissando that 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 that takes the pitch of the grace note as its argument.

Note that in polyphonic music the grace note must be matched with corresponding grace notes in all other voices.

```

repeatGliss = #(define-music-function (grace)
  (ly:pitch?)
  #{
    % the next two lines ensure the glissando is long enough
    % to be visible
    \once \override Glissando.springs-and-rods
      = #ly:spanner::set-spacing-rods
    \once \override Glissando.minimum-length = 3.5
    \once \hideNotes
    \grace $grace \glissando
  #})

\score {
  \relative c'' {
    \repeat volta 3 { c4 d e f\glissando }
    \alternative {
      { g2 d }
      { \repeatGliss f g2 e }
      { \repeatGliss f e2 d }
    }
  }
}

```

```

music = \relative c' {
  \voiceOne
  \repeat volta 2 {
    g a b c\glissando
  }
  \alternative {
    { d1 }
    { \repeatGliss c \once \omit StringNumber e1\2 }
  }
}

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

```

The image shows two staves of music. The top staff is a treble clef with a common time signature. It contains three first endings, each marked with a bracket and the number '1.' above it. The bottom staff is a guitar tablature with strings T, A, B. It contains two first endings, each marked with a bracket and the number '1.' above it. The tablature includes fingerings: 0, 2, 0, 1, 3, 5, and dots indicating repeat points.

## Flat ties

This snippet provides a function `flared-tie` to draw a tie that consist of straight lines. It is intended as a replacement for the default tie-drawing function (i.e., a replacement argument for the `stencil` property of the `Tie` grob).

The argument of `flared-tie` is a list of coordinate pairs that specify additional points between the first and last point to span up the tie's lines. The first and last point are identical to the original tie's start and end point, respectively. The X and Y coordinate values are multiples of the bounding box length and height of the original tie (also taking care of the tie's direction); consequently, the first point has coordinates (0,0), and the last point (1,0).

The function `flare-tie` defines a shorthand for a flat tie. Further tweaking of the shape is possible by overriding `Tie.details.height-limit` or with `\shape`. It is also possible to change the custom definition on the fly.

```

#(define ((flared-tie coords) grob)
  (define (pair-to-list pair)
    (list (car pair) (cdr pair)))

```



```

(define (normalize-coords goods x y dir)
  (map
    (lambda (coord)
      (cons (* x (car coord)) (* y dir (cdr coord))))
    goods))

(define (my-c-p-s points thick)
  (make-connected-path-stencil points thick 1.0 1.0 #f #f))

;; Calling `ly:tie::print` and assigning its return value to a
;; variable in this outer `let` triggers LilyPond to position the
;; tie, allowing us to extract its extents. We only proceed,
;; however, if the tie doesn't get discarded (for whatever reason).
(let ((sten (ly:tie::print grob)))
  (if (grob::is-live? grob)
      (let* ((layout (ly:grob-layout grob))
             (line-thickness (ly:output-def-lookup layout
                                                       'line-thickness))
             (thickness (ly:grob-property grob 'thickness 0.1))
             (used-thick (* line-thickness thickness))
             (dir (ly:grob-property grob 'direction))
             (xex (ly:stencil-extent sten X))
             (yex (ly:stencil-extent sten Y))
             (lenx (interval-length xex))
             (leny (interval-length yex))
             (xtrans (car xex))
             (ytrans (if (> dir 0) (car yex) (cdr yex))))
        ;; Add last point.
        (coord-list (append coords '((1.0 . 0.0))))
        (uplist
          (map pair-to-list
              (normalize-coords coord-list lenx (* leny 2) dir))))
      (ly:stencil-translate
        (my-c-p-s uplist used-thick)
        (cons xtrans ytrans)))
    '()))

% Define a default tie shape consisting of three straight lines.
#(define flare-tie
  (flared-tie '((0.1 . 0.3) (0.9 . 0.3))))

\relative c' {
  a4~ a
  \once \override Tie.stencil = #flare-tie
  a4~ a \break

  <a c e a c e a c e>~ q
  \once \override Tie.stencil = #flare-tie
  q~ q\break

  <>~\markup \small \typewriter "height-limit = 14"

```

```

\override Tie.details.height-limit = 14
a'4~ a
\once \override Tie.stencil = #flare-tie
a4~ a \break

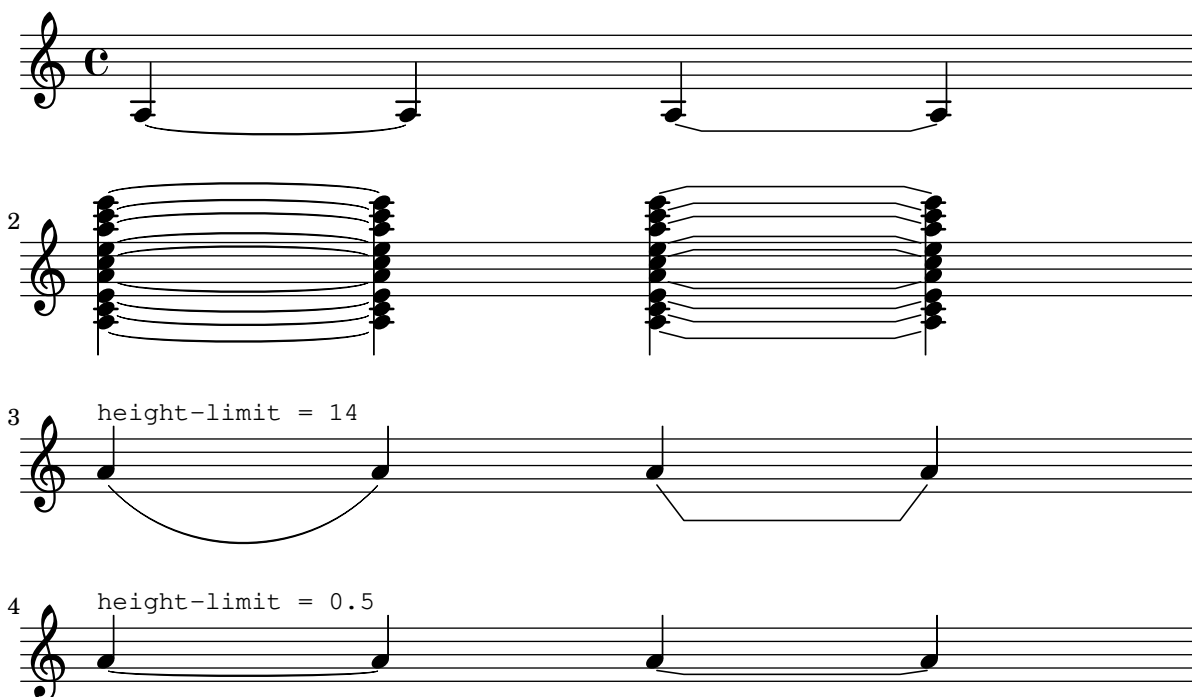
<>^\markup \small \typewriter "height-limit = 0.5"
\override Tie.details.height-limit = 0.5
a4~ a
\once \override Tie.stencil = #flare-tie
a4~ a \break

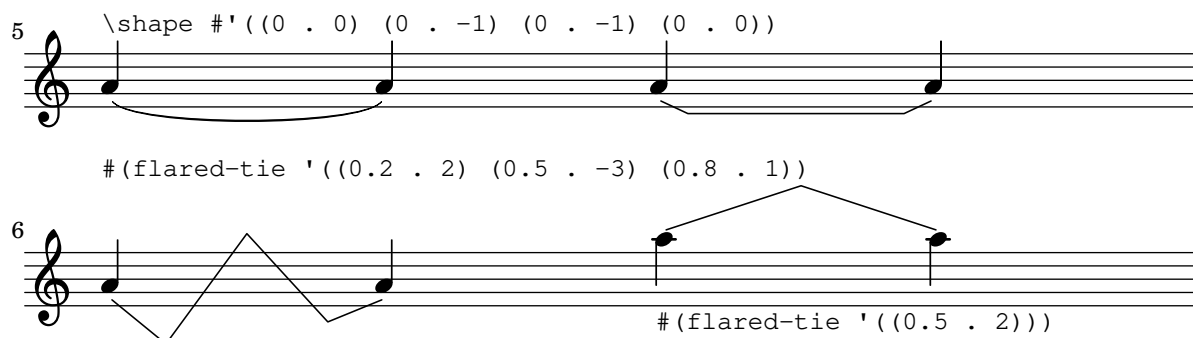
\revert Tie.details.height-limit

<>^\markup \small \typewriter
  "\shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0))"
\shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0)) Tie
a4~ a
\once \override Tie.stencil = #flare-tie
\shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0)) Tie
a4~ a \break

<>^\markup \small \typewriter
  "#(flared-tie '((0.2 . 2) (0.5 . -3) (0.8 . 1)))"
\once \override Tie.stencil =
  "#(flared-tie '((0.2 . 2) (0.5 . -3) (0.8 . 1)))"
a4~ a
<>_\markup \small \typewriter
  "#(flared-tie '((0.5 . 2)))"
\once \override Tie.stencil = #(flared-tie '((0.5 . 2)))
a'4~ a
}

```





## Flute slap notation

It is possible to indicate special articulation techniques such as a flute “tongue slap” by replacing the note head with the appropriate glyph. For that we can draw the accent-like note head with `\markup`.

```
slap =
#(define-music-function (music) (ly:music?)
  #{
    \temporary \override NoteHead.stencil =
      #ly:text-interface::print
    \temporary \override NoteHead.text =
      \markup
        \translate #'(1 . 0)
        \override #'(thickness . 1.4)
        \overlay { \draw-line #'(-1.2 . 0.4)
                  \draw-line #'(-1.2 . -0.4) }
    \temporary \override NoteHead.stem-attachment =
      #(lambda (grob)
        (let* ((stem (ly:grob-object grob 'stem))
              (dir (ly:grob-property stem 'direction UP))
              (is-up (eqv? dir UP)))
          (cons dir (if is-up 0 -0.8)))))
    #music
    \revert NoteHead.stencil
    \revert NoteHead.text
    \revert NoteHead.stem-attachment
  })

\relative c' {
  c4 \slap c d r
  \slap { g4 a } b r
}
```



## 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-);"

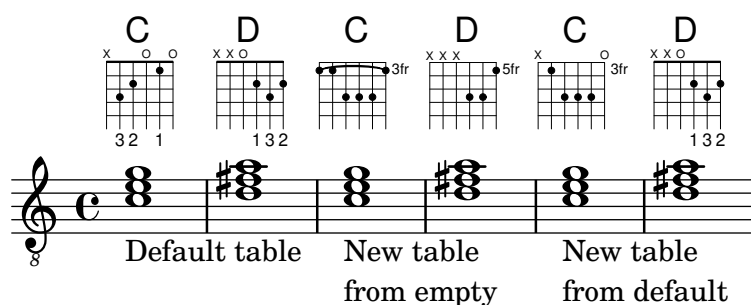
% Add a chord to `custom-fretboard-table-two`.
\storePredefinedDiagram #custom-fretboard-table-two
  \chordmode {c}
  #guitar-tuning
  "x;3;5;5;5;o;"

<<
  \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 |
}
>>
}
>>

```



## Generate special note head shapes

When a note head with a special shape cannot easily be generated with graphic markup, a drawing specification for `ly:make-stencil` can be used to generate the shape. This example gives an example for a parallelogram-shaped note head.

Unfortunately, the available commands in a drawing specification are currently not documented (<https://gitlab.com/lilypond/lilypond/-/issues/6874>); in any case, the used `path` sub-command has the following signature, quite similar to the `make-path-stencil` Scheme function.

```
(path thickness command-list line-cap-style line-join-style fill)
```

The commands in *command-list* resemble PostScript drawing commands but with arguments after the command name.

```

parallelogram =
  #(ly:make-stencil
    '(path 0.1
      (rmoveto 0 0.25
        lineto 1.2 0.75
        lineto 1.2 -0.25
        lineto 0 -0.75
        lineto 0 0.25)
      round
      round
      #t)
    (cons -0.05 1.25)
    (cons -.75 .75))

myNoteHeads = \override NoteHead.stencil = \parallelogram
normalNoteHeads = \revert NoteHead.stencil

\relative c' {
  \myNoteHeads
  g4 d'
  \normalNoteHeads
  <f, \tweak stencil \parallelogram b e>4 d
}

```



## Generating custom flags

The stencil property of the Flag grob can be set to a custom Scheme function to generate the glyph for the flag.

```
#(define-public (weight-flag grob)
  (let* ((stem-grob (ly:grob-parent grob X))
         (log (- (ly:grob-property stem-grob 'duration-log) 2))
         (is-up? (eqv? (ly:grob-property stem-grob 'direction) UP))
         (yext (if is-up? (cons (* log -0.8) 0) (cons 0 (* log 0.8))))
         (flag-stencil (make-filled-box-stencil '(-0.4 . 0.4) yext))
         (stroke-style (ly:grob-property grob 'stroke-style))
         (stroke-stencil (if (equal? stroke-style "grace")
                              (make-line-stencil 0.2 -0.9 -0.4 0.9 -0.4)
                              empty-stencil)))
    (ly:stencil-add flag-stencil stroke-stencil)))

% Create a flag stencil by looking up the glyph from the font
#(define (inverted-flag grob)
  (let* ((stem-grob (ly:grob-parent grob X))
         (dir (if (eqv? (ly:grob-property stem-grob 'direction) UP) "d" "u"))
         (flag (retrieve-glyph-flag "" dir "" grob))
         (line-thickness (ly:staff-symbol-line-thickness grob))
         (stem-thickness (ly:grob-property stem-grob 'thickness))
         (stem-width (* line-thickness stem-thickness))
         (stroke-style (ly:grob-property grob 'stroke-style))
         (stencil (if (null? stroke-style)
                      flag
                      (add-stroke-glyph flag stem-grob dir stroke-style "")))
         (rotated-flag (ly:stencil-rotate-absolute stencil 180 0 0)))
    (ly:stencil-translate rotated-flag (cons (- (/ stem-width 2)) 0))))

snippetexamplenotes =
{
  \autoBeamOff c'8 d'16 c'32 d'64 \acciaccatura {c'8} d'64
}

{
  \time 1/4
  <>^"Normal flags"
  \snippetexamplenotes

  <>_"Custom flag: inverted"
  \override Flag.stencil = #inverted-flag
  \snippetexamplenotes

  <>^"Custom flag: weight"
  \override Flag.stencil = #weight-flag
}
```

```
\snippetexamplenotes

<>_"Revert to normal"
\revert Flag.stencil
\snippetexamplenotes
}
```



## Generating random notes

This Scheme-based snippet generates random notes. Use as

```
\randomNotes n from to dur
```

to generate *n* random notes between pitches *from* and *to*, with duration *dur*.

```
randomNotes =
#(define-music-function (n from to dur)
  (integer? ly:pitch? ly:pitch? ly:duration?)
  (let ((from-step (ly:pitch-steps from))
        (to-step (ly:pitch-steps to)))
    (make-sequential-music
     (map (lambda (_)
           (let* ((step (+ from-step
                           (random (- to-step from-step))))
                 (pitch (ly:make-pitch 0 step 0)))
             #{ $pitch $dur #}))
          (iota n))))))
```

```
\randomNotes 24 c' g'' 8
```



## Generating whole scores (also book parts) in Scheme without using the parser

A LilyPond score internally is just a Scheme expression, generated by the LilyPond parser. Using Scheme, one can also automatically generate a score without an input file. If you have the music expression in Scheme, a score can be generated by simply calling

```
(scorify-music music)
```

on your music. This generates a score object, for which you can then set a custom layout block with

```
(let* ((layout (ly:output-def-clone $defaultlayout)))
  ; modify the layout here, then assign it:
  (ly:score-add-output-def! score layout))
```

Finally, all you have to do it to pass this score to LilyPond for typesetting. This snippet defines functions (add-score score), (add-text text), and (add-music music) to pass a complete score, some markup, or some music to LilyPond for typesetting.

This snippet also works for typesetting scores inside a `\book {...}` block as well as top-level scores. To achieve this, each score scheduled for typesetting is appended to the list of top-level scores, and the top-level book handler (which is a Scheme function called to process a book once a `\book{...}` block is closed) is modified to insert all collected scores so far to the book.

Note: For technical reasons, only the first `\book` is shown, as the other `\book` commands create additional output files.

```
#(define-public (add-score score)
  (ly:parser-define! 'toplevel-scores
    (cons score (ly:parser-lookup 'toplevel-scores))))

#(define-public (add-text text)
  (add-score (list text)))

#(define-public (add-music music)
  (collect-music-aux (lambda (score)
    (add-score score))
    music))

#(define-public (toplevel-book-handler book)
  (map (lambda (score)
    (ly:book-add-score! book score))
    (reverse! (ly:parser-lookup 'toplevel-scores)))
  (ly:parser-define! 'toplevel-scores (list))
  (print-book-with-defaults book))

#(define-public (book-score-handler book score)
  (add-score score))

#(define-public (book-text-handler book text)
  (add-text text))

#(define-public (book-music-handler book music)
  (add-music music))

% Some example code to show how to use these functions. Each call to
% `oneNoteScore` constructs a global markup followed by a single
% staff with a single quarter note. The pitch of this note is taken
% from the variable `pitch`; the start value 0 corresponds to pitch C.
% After emitting the score, variable `pitch` gets increased by 1.
%
% `oneNoteScore` calls Scheme function `add-one-note-score` to do all
% the work.

#(define add-one-note-score #f)
#(let ((pitch 0))
  (set! add-one-note-score
    (lambda ()
      (let* ((music
        (make-music
          'EventChord
```



```

'elements (list (make-music
                  'NoteEvent
                  'duration (ly:make-duration 2 0 1/1)
                  'pitch (ly:make-pitch 0 pitch 0))))
(score (scorify-music music))
(layout (ly:output-def-clone $defaultlayout))
(note-name (case pitch
              ((0) "do")
              ((1) "ré")
              ((2) "mi")
              ((3) "fa")
              ((4) "sol")
              ((5) "la")
              ((6) "si")
              (else "huh")))
(title (markup #:large #:line
               ("Score with a" note-name))))
(ly:score-add-output-def! score layout)
(add-text title)
(add-score score)
(set! pitch (modulo (1+ pitch) 7))))

oneNoteScore =
#(define-void-function () ()
  (add-one-note-score))

\book {
  \oneNoteScore

  \paper { tagline = ##f }
}

\book {
  \oneNoteScore
  \oneNoteScore

  \paper { tagline = ##f }
}

% Top-level scores are also handled correctly.
\oneNoteScore
\oneNoteScore

\paper { tagline = ##f }

```

Score with a do



## Isolated percent repeats

Isolated percents can also be printed.

```
makePercent =
#(define-music-function (note) (ly:music?)
  "Make a percent repeat the same length as NOTE."
  (make-music 'PercentEvent
    'length (ly:music-length note)))

\relative c' {
  \makePercent s1
}
```



## Numbers as easy note heads

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

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

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

#(set-global-staff-size 30)

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

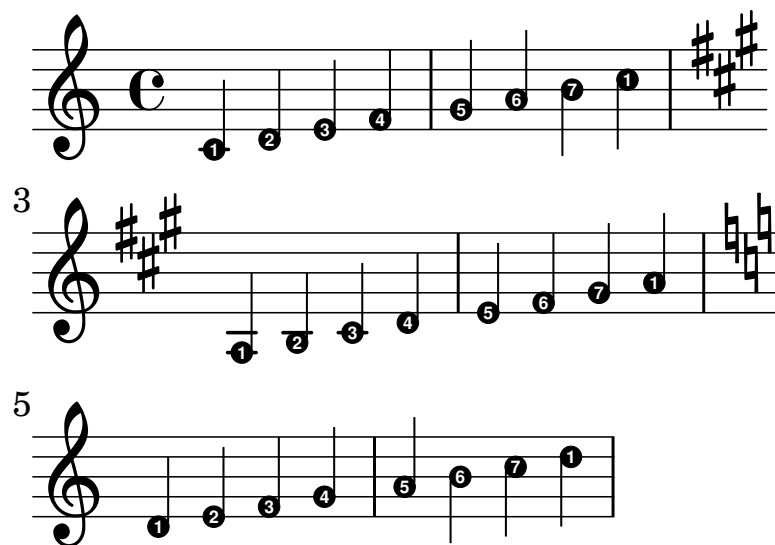
\relative c' {
  \easyHeadsOn
  c4 d e f
  g4 a b c \break
}
```

```

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

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

```



## Overriding articulations by type

Sometimes you may want to affect a single articulation type. Although it is always possible to use `\tweak`, it might become tedious to do so for every single sign of a whole score. The following shows how to tweak articulations with a list of custom settings. One use-case might be to create a style sheet.

```

#(define (custom-script-tweaks ls)
  (lambda (grob)
    (let* ((type (ly:event-property (ly:grob-property grob 'cause)
                                     'articulation-type))
           (tweaks (assoc-ref ls type)))
      (when tweaks
        (for-each
         (lambda (x) (ly:grob-set-property! grob (car x) (cdr x)))
         tweaks)))))

```

```

customScripts =
#(define-music-function (settings) (list?)
  #{
    \override Script.before-line-breaking =
      #(custom-script-tweaks settings)
  })
revertCustomScripts = \revert Script.before-line-breaking

```

% Example

```

% Predefine two sets of desired tweaks.
#(define my-settings-1
  '(
    (accent . ((font-size . 0)
                (color . (1 0 0))))
    (segno . ((font-size . 0)
              (color . (1 0 0))))
    (staccato . ((color . (1 0 0))
                 (padding . 0.5)))
    (staccatissimo . ((padding . 1)
                      (color . (1 0 0))))
    (tenuto . ((color . (1 0 0))
               (rotation . (45 0 0))
               (padding . 2)
               (font-size . 10)))
  ))

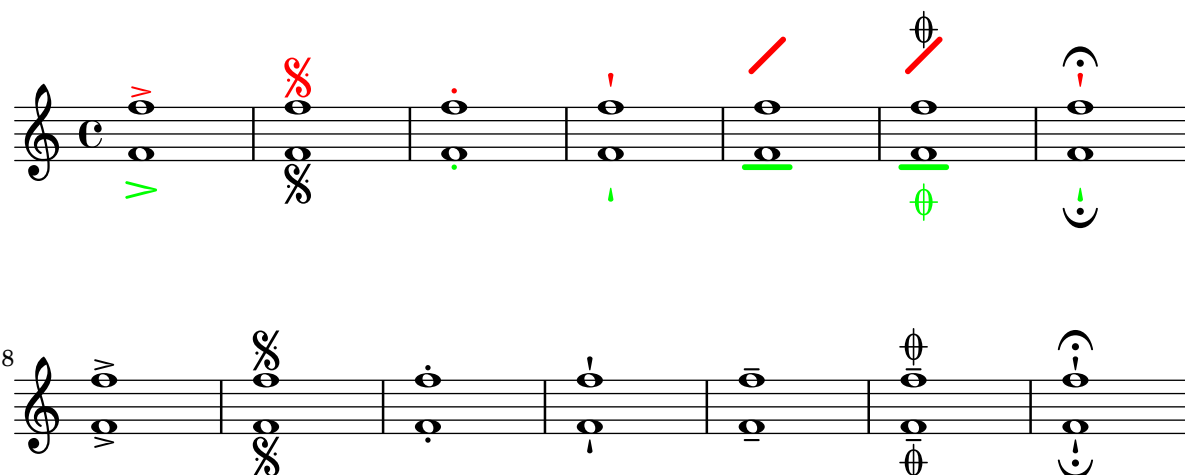
#(define my-settings-2
  '(
    (accent . ((font-size . 4)
                (color . (0 1 0))
                (padding . 1.5)))
    (coda . ((color . (0 1 0))
              (padding . 1)))
    (staccato . ((color . (0 1 0))))
    (staccatissimo . ((padding . 2)
                      (color . (0 1 0))))
    (tenuto . ((color . (0 1 0))
               (font-size . 10)))
  ))

music = { f1-> | f\segno | f-. | f-! | f-- | f--\coda | f-!\fermata | }

block = {
  \music
  \break
  \revertCustomScripts \music
}

\new Staff <<
  \new Voice \with { \customScripts #my-settings-1 }
  \relative c' { \voiceOne \block }
  \new Voice \with { \customScripts #my-settings-2 }
  \relative c' { \voiceTwo \block }
>>

```



## 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.

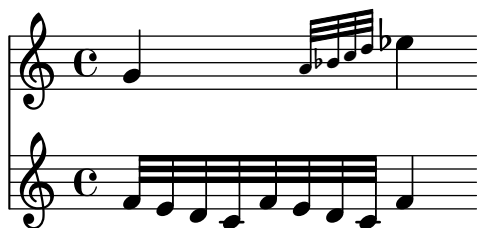
Due to issue #6876 (<https://gitlab.com/lilypond/lilypond/-/issues/6876>), however, accidentals are ignored if this property is set. This snippet gives a workaround to circumvent the problem.

Another unfortunate side effect of this property is that LilyPond does not check whether there is enough horizontal space for grace notes (this is tracked as issue #2630 (<https://gitlab.com/lilypond/lilypond/-/issues/2630>)). You have to make sure that enough space is available, for example, by using `\newSpacingSection` together with a proper value for the `base-shortest-duration` of the `SpacingSpanner` grob.

```
shiftedGrace =
#(define-music-function (offset music) (number? ly:music?)
  #{
    \override NoteHead.X-offset = #(- offset 0.85)
    \override Stem.X-offset = #offset
    \grace { $music }
    \revert NoteHead.X-offset
    \revert Stem.X-offset
  #})

\relative c' ' <<
  { g4 \shiftedGrace #-1.3 a32 \shiftedGrace #-0.5 { bes c d } es4 }
  { f,32 e d c f e d c f4 }
>>

\layout {
  \context {
    \Score
    \override SpacingSpanner.strict-grace-spacing = ##t
  }
}
```



## Print chord names with same root and different bass as slash and bass note

To print subsequent ChordNames only differing in its bass note as slash and bass note, use the Scheme engraver defined in this snippet. The behaviour may be controlled in detail by the chordChanges context property.

```
#(define Bass_changes_equal_root_engraver
  (lambda (ctx)
    "For sequential `ChordNames` with the same root but a different bass,
    the root markup is dropped: D D/C D/B -> D /C /B.
    The behaviour may be controlled by setting the `chordChanges` context
    property."
    (let ((chord-pitches '())
          (last-chord-pitches '())
          (bass-pitch #f))
      (make-engraver
        ((initialize this-engraver)
         (let ((chord-note-namer (ly:context-property ctx
                                                    'chordNoteNamer)))
           ;; Set 'chordNoteNamer, respect user setting if already done
           (ly:context-set-property! ctx 'chordNoteNamer
                                     (if (procedure? chord-note-namer)
                                         chord-note-namer
                                         note-name->markup))))
        (listeners
         ((note-event this-engraver event)
          (let* ((pitch (ly:event-property event 'pitch))
                 (pitch-name (ly:pitch-notename pitch))
                 (pitch-alt (ly:pitch-alteration pitch))
                 (bass (ly:event-property event 'bass #f))
                 (inversion (ly:event-property event 'inversion #f)))
            ;; Collect notes of the chord
            ;; - to compare inversed chords we need to collect the
            ;;   bass note as usual member of the chord, whereas an
            ;;   added bass must be treated separate from the usual
            ;;   chord-notes
            ;; - notes are stored as pairs containing their
            ;;   pitch-name (an integer), i.e. disregarding their
            ;;   octave and their alteration
            (cond (bass (set! bass-pitch pitch))
                  (inversion
                   (set! bass-pitch pitch)
                   (set! chord-pitches
```

```

        (cons (cons pitch-name pitch-alt)
              chord-pitches)))
      (else
       (set! chord-pitches
              (cons (cons pitch-name pitch-alt)
                    chord-pitches))))))

(acknowledgers
 ((chord-name-interface this-engraver grob source-engraver)
  (let ((chord-changes (ly:context-property ctx
                                             'chordChanges #f)))
    ;; If subsequent chords are equal apart from their bass,
    ;; reset the 'text-property.
    ;; Equality is done by comparing the sorted lists of this
    ;; chord's elements and the previous chord. Sorting is
    ;; needed because inverted chords may have a different
    ;; order of pitches. `chord-changes` needs to be true.
    (if (and bass-pitch
              chord-changes
              (equal?
               (sort chord-pitches car<)
               (sort last-chord-pitches car<)))
        (ly:grob-set-property!
         grob 'text
         (make-line-markup
          (list
           (ly:context-property ctx 'slashChordSeparator)
           ((ly:context-property ctx 'chordNoteNamer)
            bass-pitch)
           (ly:context-property ctx
                                'chordNameLowercaseMinor))))))
      (set! last-chord-pitches chord-pitches)
      (set! chord-pitches '())
      (set! bass-pitch #f))))

((finalize this-engraver)
 (set! last-chord-pitches '()))))

myChords = \chordmode {
  % \germanChords

  \set chordChanges = ##t
  d2:m d:m/cis

  d:m/c
  \set chordChanges = ##f
  d:m/b

  e1:7
  \set chordChanges = ##t
  e

```

```

\break

\once \set chordChanges = ##f
e1/f
e2/gis e/+gis e e:m/f d:m d:m/cis d:m/c
\set chordChanges = ##f
d:m/b
}

<<
\new ChordNames
  \with { \consists #Bass_changes_equal_root_engraver }
  \myChords
\new Staff \myChords
>>

```

The image displays two staves of musical notation. The first staff contains six measures with the following chord symbols: Dm, /C#, /C, Dm/B, E<sup>7</sup>, and E. The second staff contains eight measures with the following chord symbols: E/F, /G#, E, Em/F, Dm, /C#, /C, and Dm/B. Each chord is represented by a vertical line with a chord symbol above it and a corresponding chord symbol below it.

## Replacing default MIDI instrument equalization

The default MIDI instrument equalizer can be replaced by setting the `instrumentEqualizer` property in the `Score` context to a user-defined Scheme procedure that uses a MIDI instrument name as its argument along with a pair of fractions indicating the minimum and maximum volumes, respectively, to be applied to that specific instrument.

The following example sets the minimum and maximum volumes for flute and clarinet.

```

#(define my-instrument-equalizer-alist '())

#(set! my-instrument-equalizer-alist
  (append
    '(("flute" . (0.7 . 0.9))
      ("clarinet" . (0.3 . 0.6)))
    my-instrument-equalizer-alist))

#(define (my-instrument-equalizer s)
  (let ((entry (assoc s my-instrument-equalizer-alist)))
    (if entry
      (cdr entry))))

\score {
  <<
    \new Staff {
      \key g \major
      \time 2/2

```



```

\set Score.instrumentEqualizer = #my-instrument-equalizer
\set Staff.midiInstrument = "flute"
\new Voice \relative {
  r2 g'\mp g fis~
  4 g8 fis e2~
  4 d8 cis d2
}
}
\new Staff {
  \key g \major
  \set Staff.midiInstrument = "clarinet"
  \new Voice \relative {
    b'1\p a2. b8 a
    g2. fis8 e
    fis2 r
  }
}
>>
\layout { }
\midi { }
}

```



## Separating key cancellations from key signature changes

By default, the accidentals used for key cancellations are placed adjacent to those for key signature changes. This behavior can be changed by overriding the `break-align-orders` property of the `BreakAlignment` grob.

The value of `break-align-orders` is a vector of length 3, with quoted lists of breakable items as elements. Each list describes the default order of prefatory matter at the end, in the middle, and at the beginning of a line, respectively. We are only interested in changing the behaviour in the middle of a line.

If you look up the definition of `break-align-orders` in LilyPond's Internal Reference (see the `BreakAlignment` (<https://lilypond.org/doc/v2.24/Documentation/internals/breakalignment>) grob), you get the following order in the second element:

```

...
staff-bar
key-cancellation
key-signature
...

```

We want to change that, moving `key-cancellation` before `staff-bar`. To make this happen we use the `grob-transformer` function, which gives us access to the original vector as the second argument of the lambda function, here called *orig* (we don't need the first argument, *grob*). We return a new vector, with unchanged first and last elements. For the middle element, we first remove `key-cancellation` from the list, then adding it again before `staff-bar`.

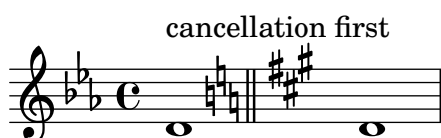
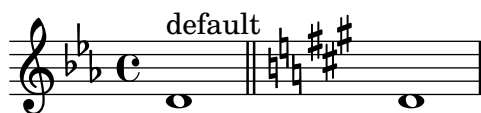
```
#(define (insert-before where what lst)
  (cond
    ((null? lst)           ; If the list is empty,
     (list what))         ; return a single-element list.
    ((eq? where (car lst)) ; If we find symbol `where`,
     (cons what lst))     ; insert `what` before curr. position.
    (else                  ; Otherwise keep building the list by
     (cons (car lst)      ; adding the current element and
           (insert-before ; recursing with the next element.
             where what (cdr lst))))))
```

```
cancellationFirst =
\override Score.BreakAlignment.break-align-orders =
#(grob-transformer
 'break-align-orders
 (lambda (grob orig)
  (let* ((middle (vector-ref orig 1))
        (middle (delq 'key-cancellation middle))
        (middle (insert-before
                        'staff-bar 'key-cancellation middle)))
    (vector
     ;; end of line
     (vector-ref orig 0)
     ;; middle of line
     middle
     ;; beginning of line
     (vector-ref orig 2)))))
```

```
music = { \key es \major d'1 \bar "||"
          \key a \major d'1 }
```

```
{ <>^\markup "default"
  \music }
```

```
{ <>^\markup "cancellation first"
  \cancellationFirst
  \music }
```



## String number extender lines

Make an extender line for string number indications, showing that a series of notes is supposed to be played all on the same string.

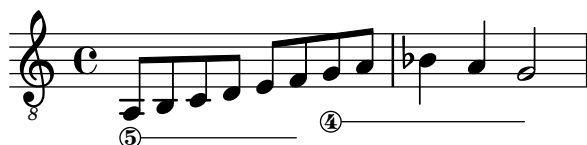
```
stringNumberSpanner =
  #(define-music-function (StringNumber) (string?)
```

```

#{
  \override TextSpanner.style = #'solid
  \override TextSpanner.font-size = #-5
  \override TextSpanner.bound-details.left.stencil-align-dir-y = #CENTER
  \override TextSpanner.bound-details.left.text =
    \markup { \circle \number $StringNumber }
  #})

\relative c {
  \clef "treble_8"
  \textSpannerDown
  \stringNumberSpanner "5" a8\startTextSpan b c d
  e f\stopTextSpan \stringNumberSpanner "4" g\startTextSpan a |
  bes4 a g2\stopTextSpan
}

```



## Three-sided box

This example shows how to add a markup command to get a three-sided box around some text (or other markup).

```

% New command to add a three-sided box, with sides north, west, and south.
% Based on the `box-stencil` command defined in `scm/stencil.scm`.
% Note that ";;" is used to comment a line in Scheme.
#(define-public (NWS-box-stencil stencil thickness padding)
  "Add a box around STENCIL, producing a new stencil."
  (let* ((x-ext (interval-widen (ly:stencil-extent stencil X) padding))
        (y-ext (interval-widen (ly:stencil-extent stencil Y) padding))
        (y-rule (make-filled-box-stencil (cons 0 thickness) y-ext))
        (x-rule (make-filled-box-stencil
                  (interval-widen x-ext thickness) (cons 0 thickness))))
    ;; (set! stencil (ly:stencil-combine-at-edge stencil X 1 y-rule padding))
    (set! stencil (ly:stencil-combine-at-edge stencil X LEFT y-rule padding))
    (set! stencil (ly:stencil-combine-at-edge stencil Y UP x-rule 0.0))
    (set! stencil (ly:stencil-combine-at-edge stencil Y DOWN x-rule 0.0))
    stencil))

% The corresponding markup command, based on the `\\box` command defined
% in `scm/define-markup-commands.scm`.
#(define-markup-command (NWS-box layout props arg) (markup?)
  #:properties ((thickness 0.1) (font-size 0) (box-padding 0.2))
  "Draw a box round ARG.

```

Look at THICKNESS, BOX-PADDING, and FONT-SIZE properties to determine line thickness and padding around the markup."

```

  (let ((pad (* (magstep font-size) box-padding))
        (m (interpret-markup layout props arg)))
    (NWS-box-stencil m thickness pad)))

```

```
\relative c' {
  c1^\markup { \NWS-box ABCD }
  c1^\markup { \NWS-box \note {4} #1.0 }
}
```



## 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))
        ;; `ly:pitch-alteration` returns quarter tone steps.
        (a (* 4 (ly:pitch-alteration p)))
        (n (ly:pitch-notename p)))
    (cond
      ((and (> a 1)
            (or (eqv? n 6) (eqv? n 2))))
      (set! a (- a 2))
      (set! n (+ n 1)))
      ((and (< a -1)
            (or (eqv? n 0) (eqv? 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))))
      (when (< n 0)
        (set! o (- o 1))
        (set! n (+ n 7)))
      (when (> n 6)
        (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)))
    (when (pair? es)
      (ly:music-set-property! music 'elements
                              (map naturalize es)))

    (when (ly:music? e)
      (ly:music-set-property! music 'element
                              (naturalize e)))

    (when (ly:pitch? p)
      (set! p (naturalize-pitch p))
      (ly:music-set-property! music 'pitch p))
    music))

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

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

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

```



## Two \partCombine pairs on one staff

The `\partCombine` function takes two music expressions, each containing a part, and distributes them among four Voice contexts named “one”, “two”, “solo”, and “shared”, depending on when and how the parts are merged into a common voice.

Variants of `\partCombine` are `\partCombineUp` and `\partCombineDown` to produce up-stem and down-stem merging of two voices, respectively. Combining them to squeeze four parts into a single staff, however, need some special setup, which this snippet defines accordingly.

```

customPartCombineUp =
#(define-music-function (part1 part2) (ly:music? ly:music?)
  "Make an up-stem `VoiceBox` context that combines PART1 and PART2.

```

The context is called 'Up'; internally, the function calls `\partCombineUp`.`

```

#{
  \new VoiceBox = "Up" <<
    \context Voice = "one" { \voiceOne }
    \context Voice = "two" { \voiceThree }

```

```

\context Voice = "shared" { \voiceOne }
\context Voice = "solo" { \voiceOne }
\context NullVoice = "null" {}
\partCombine #part1 #part2
>>
#})

```

```

customPartCombineDown =
#(define-music-function (part3 part4) (ly:music? ly:music?)
  "Make a down-stem `VoiceBox` context that combines PART3 and PART4.

```

The context is called 'Down'; internally, the function calls  
`\\partCombineDown`.`

```

#{
  \new VoiceBox = "Down" <<
    \set VoiceBox.soloText = #"Solo III"
    \set VoiceBox.soloIIIText = #"Solo IV"
    \context Voice = "one" { \voiceFour }
    \context Voice = "two" { \voiceTwo }
    \context Voice = "shared" { \voiceFour }
    \context Voice = "solo" { \voiceFour }
    \context NullVoice = "null" {}
    \partCombine #part3 #part4
  >>
#})

```

```

soprano = { d'4 | cis' b e' d'8 cis' | cis'2 b }
alto = { fis4 | e8 fis gis ais b4 b | b ais fis2 }
tenor = { a8 b | cis' dis' e'4 b8 cis' d'4 | gis cis' dis'2 }
bass = { fis8 gis | a4 gis g fis | eis fis b,2 }

```

```

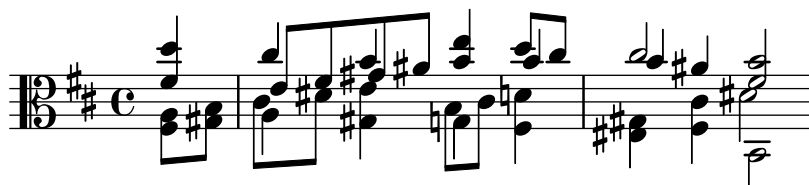
\new Staff <<
  \key b\minor
  \clef alto
  \partial 4
  \transpose b b' \customPartCombineUp \soprano \alto
  \customPartCombineDown \tenor \bass
>>

```

```

\layout {
  \context {
    \Staff
    \accepts "VoiceBox"
  }
  \context {
    \name "VoiceBox"
    \type "Engraver_group"
    \defaultchild "Voice"
    \accepts "Voice"
    \accepts "NullVoice"
  }
}

```



## User-defined time signatures

New time signature styles can be defined. The time signature in the second measure is printed upside down in both staves.

```
#(add-simple-time-signature-style 'topsy-turvy
  (lambda (fraction)
    (make-rotate-markup 180 (make-compound-meter-markup fraction))))
```

```
<<
  \new Staff {
    \time 3/4 f'2.
    \override Score.TimeSignature.style = #'topsy-turvy
    \time 3/4 R2. \bar "|"
  }
  \new Staff {
    R2. e''
  }
>>
```



## Using ly:grob-object to access grobs with \tweak

Some grobs can be accessed “laterally” from within another grob’s callback. These are usually listed as “layout objects” in the “Internal properties” section of a grob interface. The function `ly:grob-object` is used to access these grobs.

Demonstrated below are some ways of accessing grobs from within a `NoteHead` callback, but the technique is not limited to `NoteHeads`. However, the `NoteHead` callback is particularly important, since it is the implicit callback used by the `\tweak` command.

The console output of the example function below (`display-grobs`) is as follows.

```
-----
#<Grob Accidental >
()
#<Grob Stem >
```

It is probably not that useful, but it demonstrates that the grobs are indeed being accessed.

```
#(define (notehead-get-accidental notehead)
  ;; notehead is grob
  (ly:grob-object notehead 'accidental-grob))
```

```
#(define (notehead-get-arpeggio notehead)
```

```

;; notehead is grob
(let ((notecolumn (notehead-get-notecolumn notehead)))
  (ly:grob-object notecolumn 'arpeggio)))

#(define (notehead-get-notecolumn notehead)
  ;; notehead is grob
  (ly:grob-parent notehead X))

#(define (notehead-get-stem notehead)
  ;; notehead is grob
  (let ((notecolumn (notehead-get-notecolumn notehead)))
    (ly:grob-object notecolumn 'stem)))

#(define (display-grobs notehead)
  ;; notehead is grob
  (let ((accidental (notehead-get-accidental notehead))
        (arpeggio (notehead-get-arpeggio notehead))
        (stem (notehead-get-stem notehead)))
    (format (current-error-port) "~2&~a\n" (make-string 20 #\ -))
    (for-each
     (lambda (x) (format (current-error-port) "~a\n" x))
     (list accidental arpeggio stem))))

\relative c' {
  %% display grobs for each note head:
  \%override NoteHead.before-line-breaking = #display-grobs
  <c
  %% or just for one:
  \tweak before-line-breaking #display-grobs
  es
  g>1\arpeggio
}

```



## Vertical line as a baroque articulation mark

This short vertical line placed above the note is commonly used in baroque music. Its meaning can vary, but generally indicates notes that should be played with more “weight”. The following example demonstrates how to achieve such a notation.

```

upline =
\tweak stencil
#(lambda (grob)
  (grob-interpret-markup grob #{ \markup \draw-line #'(0 . 1) #}))
\stopped

\relative c' {
  a'4^\upline a( c d')_\upline
}

```





## 33 Spacing

See also Section “Spacing issues” in *Notation Reference*.

### Adjusting vertical spacing of lyrics

This snippet shows how to bring the lyrics line closer to the staff.

*% Default layout:*

<<

```
\new Staff \new Voice = melody \relative c' {
  c4 d e f | g4 f e d | c1
}
\new Lyrics \lyricsto melody { aa aa aa aa aa aa aa aa aa }
```

```
\new Staff \new Voice = melody \relative c' {
  c4 d e f | g4 f e d | c1
}
```

*% Reducing the minimum space below the staff and above the lyrics:*

```
\new Lyrics \with {
  \override VerticalAxisGroup.nonstaff-relatedstaff-spacing =
    #'((basic-distance . 1))
}
```

```
\lyricsto melody { aa aa aa aa aa aa aa aa aa }
```

>>



### Allowing fingerings to be printed inside the staff

By default, vertically oriented fingerings are positioned outside the staff; that behavior, however, may be disabled. Attention needs to be paid to situations where fingerings and stems are in the same direction: by default, fingerings will avoid only beamed stems. That setting can be changed to avoid no stems or all stems; the following example demonstrates these two options, as well as how to go back to the default behavior.

```
\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
  a8[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = ##f
  a[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = ##t
  a[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = #only-if-beamed
  a[-1 b]-2 g-0 r
```

}

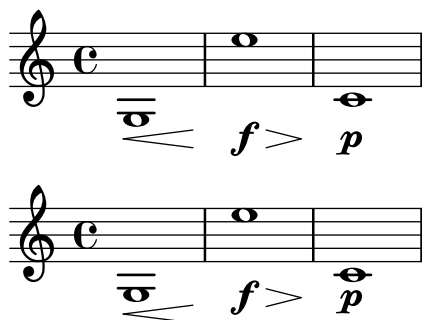


## Breaking vertical alignment of dynamics and textscripts

By default, LilyPond uses `DynamicLineSpanner` grobs to vertically align successive dynamic objects like hairpins and dynamic text. However, this is not always wanted. By inserting `\breakDynamicSpan`, which ends the alignment spanner prematurely, this vertical alignment can be avoided.

```
{ g1\< |
  e''\f\> |
  c'\p }
```

```
{ g1\< |
  e''\breakDynamicSpan\f\> |
  c'\p }
```



## Page label

Page labels may be placed inside music or at top-level, and referred to in markups.

```
\label license
```

```
\markup "This snippet is available under the Creative Commons
Public Domain Dedication license."
```

```
{
  \repeat volta 2 {
    \label startRepeat
    \repeat unfold 20 { c'2 2 }
    \pageBreak
    2 2
  }
  \textEndMark \markup {
    \with-link #'startRepeat \line {
      To page \page-ref #'startRepeat "0" "?"
    }
  }
}
```

```
\markup {
  See page \page-ref #'license "0" "?" for
  licensing information.
}
```

This snippet is available under the Creative Commons Public Domain Dedication license.

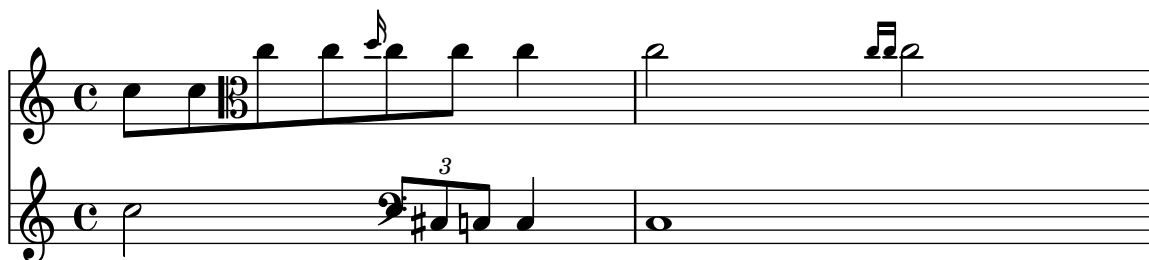
The image shows a musical score snippet with six staves. Each staff contains a sequence of eighth notes. The staves are numbered 1, 5, 9, 13, 17, and 21. The music ends with a double bar line and repeat dots. The text "To page ?" is written above the final staff.

See page ? for licensing information.

## Proportional strict notespacing

If strict-note-spacing is set spacing of notes is not influenced by bars or clefs within a system. Rather, they are placed just before the note that occurs at the same time. This may cause collisions.

```
\relative c' ' <<
  \override Score.SpacingSpanner.strict-note-spacing = ##t
  \set Score.proportionalNotationDuration = #1/16
  \new Staff {
    c8[ c \clef alto c c \grace { d16 } c8 c] c4
    c2 \grace { c16[ c16] } c2
  }
  \new Staff {
    c2 \tuplet 3/2 { c8 \clef bass cis,, c } c4
    c1
  }
  >>
```



## Reconciling bar lines thickness when staves are of different sizes (TODO duplicates snippet in Documentation/en/notation/spacing.itely)

A regularly occurring problem in ensemble or chamber music scores, particularly with piano, is that when some of the staves are printed in a smaller size, their bar lines do not match those of larger staves. This may become particularly egregious for closing double bar lines, as demonstrated here.

There are different solutions, which all involve correcting some of the properties overridden by the `\magnifyStaff` command: namely, `kern thick-thickness` and `hair-thickness` (the latter of which also governs ordinary bar lines). These are properties of the `BarLine` graphical object (“*grob*”), which resides in the `Staff` context.

This can be done by using either the `\revert` command, or more advanced commands such as `revert-props`. Alternatively, some effects of `\magnifyStaff` may be replicated on other staves through the `scale-props` command, which may be used to harmonize all bar lines either to their appearance on the smallest staves, or to some intermediate value.

*%% This snippet relies on some internal commands that are not part of the  
%% most common and stable syntax generally recommended to regular users.*

```
\markup \fontsize #-3 {
  \center-column {
    \vspace #0.3
    Default:
    \vspace #1.7
    \score {
      \new StaffGroup <<
        \new Staff \with {
          \magnifyStaff #1/2
        } { b1 b \bar "|."}
        \new Staff { b b }
      >>
    }
  }
}
\hspace #2
\center-column {
  "Reverting only the"
  "final bar line:"
  \vspace #1.2
  \score {
    \new StaffGroup <<
      \new Staff \with {
        \magnifyStaff #1/2
      } {
```

```

        b1 b
        \revert Staff.BarLine.thick-thickness
        \revert Staff.BarLine.hair-thickness
        \revert Staff.BarLine.kern
        \bar "|"
    }
    \new Staff { b b }
  >>
}
}
\hspace #2
\center-column {
  \line {Cancelling \typewriter "\magnifyStaff"}
  "only for bar lines:"
  \vspace #1.1
  \score {
    \new StaffGroup <<
      \new Staff \with {
        \magnifyStaff #1/2
        \revert-props 'magnifyStaff 0
                      '((BarLine thick-thickness)
                        (BarLine hair-thickness)
                        (BarLine kern)))
      } { b1 b \bar "|" }
    \new Staff { b b }
  >>
}
}
\hspace #2
\center-column {
  \line {Mimicking \typewriter "\magnifyStaff"}
  "on the other staves:"
  \vspace #1.3
  \score {
    \new StaffGroup <<
      \new Staff \with {
        \magnifyStaff #1/2
      } { b1 b \bar "|" }
      \new Staff \with {
        \scale-props 'magnifyStaff 1/2 #t
                      '((BarLine thick-thickness)
                        (BarLine hair-thickness)
                        (BarLine kern)))
      } { b b }
    >>
  }
}
}
\hspace #2
\center-column {
  "Applying an intermediate"
  "value to all staves:"
  \vspace #1.3

```

```

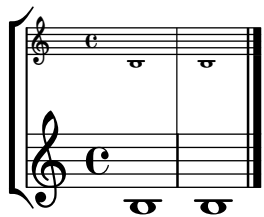
\score {
  \new StaffGroup <<
    \new Staff \with {
      \magnifyStaff #1/2
      #(scale-props 'magnifyStaff 3/2 #t
        '((BarLine thick-thickness)
          (BarLine hair-thickness)
          (BarLine kern)))
    } { b1 b \bar "|" }
    \new Staff \with {
      #(scale-props 'magnifyStaff 3/4 #t
        '((BarLine thick-thickness)
          (BarLine hair-thickness)
          (BarLine kern)))
    } { b b }
  >>
}
\layout { indent = 0 }

```

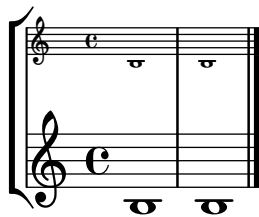
Default:



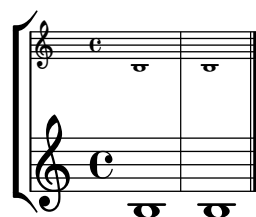
Reverting only the final bar line:



Cancelling \magnifyStaff only for bar lines:



Mimicking \magnifyStaff on the other staves:



Applying an intermediate value to all staves:



## Vertically aligned dynamics and textscripts

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

The same idea, together with `\textLengthOn`, is used to align the text scripts along their baseline.

```

music = \relative c' {
  a'2\p b\f
  e4\p f\f\> g, b\p
  c2^\markup { \huge gorgeous } c^\markup { \huge fantastic }
}

{
  \music
  \break
  \override DynamicLineSpanner.staff-padding = 3
  \textLengthOn
  \override TextScript.staff-padding = 1
  \music
}

```

```
\paper { tagline = ##f }
```



## Vertically aligning ossia and lyrics

This snippet demonstrates the use of the context properties `alignBelowContext` and `alignAboveContext` to control the positioning of lyrics and ossia.

```
\relative c' <<
  \new Staff = "1" { c4 c c c }
  \new Staff = "2" { d4 d d d }
  \new Staff = "3" { e4 e e e }

  { \skip 2
    <<
      \lyrics {
        \set alignBelowContext = "1"
        lyrics4 below
      }
      \new Staff \with {
        alignAboveContext = "3"
        fontSize = -2
        \override StaffSymbol.staff-space = #(magstep -2)
        \remove "Time_signature_engraver"
        \override VerticalAxisGroup.staff-staff-spacing =
          #'((minimum-distance . 0)
             (basic-distance . 0)
             (padding . 1))
      } {
        \tuplet 6/4 {
          \override TextScript.padding = 2
          c8[~"ossia above" d e d e f]
        }
      }
    }
  }
>>
```



A musical score example illustrating the placement of lyrics and ossia notation. The score consists of three staves, each with a treble clef and a common time signature (C). The first staff contains five quarter notes. The second staff also contains five quarter notes, with the text "lyrics below" centered between the first and second staves. The third staff contains five quarter notes, with the text "ossia above" centered between the second and third staves. Above the third staff, there is a small musical staff with a treble clef and a common time signature, containing six eighth notes beamed together, with a "6" above it.

## 34 Specific notation

### Accordion register symbols

Accordion register symbols are available as `\markup` as well as as standalone music events (as register changes tend to occur between actual music events). Bass registers are not overly standardized. The available commands can be found in ‘Discant symbols’ in the Notation Reference (<https://lilypond.org/doc/v2.24/Documentation/notation/accordion#discant-symbols>).

```
#(use-modules (lily 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" }
    >>
  }
>>
```

### 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 = #'(-1 . 3)
  \consists "Bar_engraver"
}

\chordmode {
  f1:maj7 f:7 bes:7
}

```

$F^{\Delta} \quad | \quad F^7 \quad | \quad B\flat^7 \quad |$

## Adding drum parts

Using the powerful pre-configured tools such as the `\drummode` function and the `DrumStaff` context, inputting drum parts is quite easy: drums are placed at their own staff positions (with a special clef symbol) and have note heads according to the drum. Attaching an extra symbol to the drum or restricting the number of lines is possible.

```

drh = \drummode {
  cymc4.^"crash" hhc16^"h.h." hh hhc8 hho hhc8 hh16 hh
  hhc4 r4 r2
}
drl = \drummode {
  bd4 sn8 bd bd4 << bd ss >>
  bd8 tommh tommh bd toml toml bd tomfh16 tomfh
}
timb = \drummode {
  timh4 ssh timl8 ssh r timh r4
  ssh8 timl r4 cb8 cb
}

\score {
  <<
    \new DrumStaff \with {
      instrumentName = "timbales"
      drumStyleTable = #timbales-style
      \override StaffSymbol.line-count = #2
      \override BarLine.bar-extent = #'(-1 . 1)
    }
    <<
      \timb
    >>
    \new DrumStaff \with { instrumentName = "drums" }
    <<
      \new DrumVoice { \stemUp \drh }
      \new DrumVoice { \stemDown \drl }
    >>
  >>
  \layout { }
  \midi { \tempo 4 = 120 }
}

```

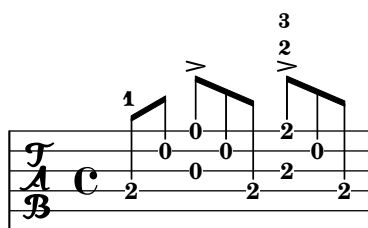


## Adding fingerings to tablatures

To add fingerings to tablatures, use a combination of `\markup` and `\finger`.

```
one = \markup { \finger 1 }
two = \markup { \finger 2 }
threeTwo = \markup {
  \override #'(baseline-skip . 2)
  \column {
    \finger 3
    \finger 2
  }
}

\score {
  \new TabStaff {
    \tabFullNotation
    \stemUp
    e8\4~\one b\2 <g\3 e'\1>~>[ b\2 e\4]
    <a\3 fis'\1>~>~\threeTwo[ b\2 e\4]
  }
}
```



## Aiken head thin variant noteheads

Aiken head white notes get harder to read at smaller staff sizes, especially with ledger lines. Losing interior white space makes them appear as quarter notes.

```
\score {
  {
    \aikenHeads
    c''2 a' c' a

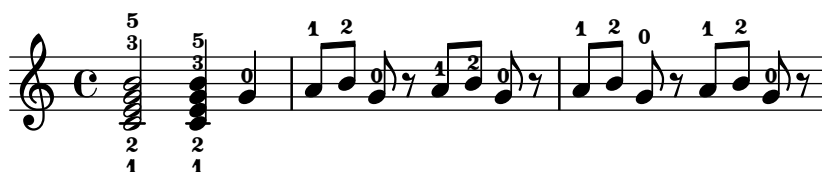
    % Switch to thin-variant noteheads
    \set shapeNoteStyles = ##(doThin reThin miThin
                          faThin sol laThin tiThin)
    c'' a' c' a
  }
}
```



## Allowing fingerings to be printed inside the staff

By default, vertically oriented fingerings are positioned outside the staff; that behavior, however, may be disabled. Attention needs to be paid to situations where fingerings and stems are in the same direction: by default, fingerings will avoid only beamed stems. That setting can be changed to avoid no stems or all stems; the following example demonstrates these two options, as well as how to go back to the default behavior.

```
\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
  a8[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = ##f
  a[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = ##t
  a[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = #only-if-beamed
  a[-1 b]-2 g-0 r
}
```



## Changing the number of lines in a staff

The number of lines in a staff may be changed by overriding the `StaffSymbol` property `line-count`.

```
upper = \relative c'' {
  c4 d e f
}

lower = \relative c {
  \clef bass
  c4 b a g
}

\score {
  \context PianoStaff <<
    \new Staff {
      \upper
    }
    \new Staff {
      \override Staff.StaffSymbol.line-count = #4
      \lower
    }
  >>
}
```



## Chant or psalm notation

This form of notation is used for psalm chant, where verses aren't always of the same length.

```
stemOff = \hide Staff.Stem
```

```
stemOn = \undo \stemOff
```

```
\score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \key g \minor
    \cadenzaOn
    \stemOff a'\breve bes'4 g'4
    \stemOn a'2 \section
    \stemOff a'\breve g'4 a'4
    \stemOn f'2 \section
    \stemOff a'\breve~\markup { \italic flexe }
    \stemOn g'2 \fine
  }
}
```



## Chord name exceptions

The property `chordNameExceptions` stores a list of chord name exceptions to handle cases either not covered or handled incorrectly.

The default chord names used by LilyPond follow the rules as given in Klaus Ignatzek's book "Die Jazzmethode für Klavier 1"; the algorithm to convert chords to chord names can be found in file `scm/chord-ignatzek-names.scm`. Additional rules are given as chord exceptions and stored in the variable `ignatzekExceptions`, as set up in file `ly/chord-modifiers-init.ly`.

This snippet modifies these exceptions in three steps.

1. Set up some music with chords and associated markup. By convention, the root (i.e., the lowest note) of each chord should have pitch `c`.
2. Call Scheme function `sequential-music-to-chord-exceptions` to create a new list of exceptions, then concatenate it with the existing ones. Since `ignatzekExceptions` is set up with this function's second parameter set to `#t` (to ignore the root of the chords), we have to do the same.
3. Register the new exception list.

```
% Step 1: Define music with chords and markup for maj9 and 6(add9).
```

```
chExceptionMusic = {
  <c e g b d'>-\markup { \super "maj9" }
  <c e g a d'>-\markup { \super "6(add9)" }
```

```

}

% Step 2: Create extended exception list.
chExceptions =
#(append (sequential-music-to-chord-exceptions chExceptionMusic #t)
         ignatzekExceptions)

theMusic = \chordmode {
  g1:maj9 g1:6.9
  % Step 3: Register extended exception list.
  \set chordNameExceptions = #chExceptions
  g1:maj9 g1:6.9
}

\layout {
  ragged-right = ##t
}

<<
  \new ChordNames \theMusic
  \new Voice \theMusic
>>

```



## Chord name major7

The layout of the major 7 can be tuned with `majorSevenSymbol`.

```

\chords {
  c:7+
  \set majorSevenSymbol = \markup { j7 }
  c:7+
}

```

$C^\Delta C^{j7}$

## Chords with stretched fingering for FretBoards and TabVoice

Sometimes chords with a stretched fingering are required. If not otherwise specified the context property `maximumFretStretch` is set to value 4, though, resulting in a warning about “No string for pitch ...”, and the note is omitted. You may set `maximumFretStretch` to an appropriate value or explicitly assign string numbers to all notes of a chord to fix that.

```

% The code below prints two warnings for the second chord,
% which may be omitted by uncommenting the following line.
%
% #(for-each (lambda (x) (ly:expect-warning "No string for pitch")) (iota 2))

mus = {
  <c' bes'>
  <c'\2 bes'>
}

```

```

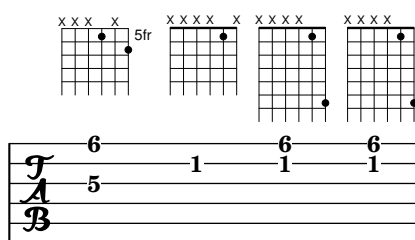
\set maximumFretStretch = 5
<c' bes'>
<c'\2 bes'\1>
}

```

```

<<
\new FretBoards \mus
\new TabVoice \mus
>>

```



## Clusters

Clusters are a device to denote that a complete range of notes is to be played.

```

fragment = \relative c' {
  c4 f <e d'>4
  <g a>8 <e a> a4 c2 <d b>4
  e2 c
}

```

```

<<
\new Staff \fragment
\new Staff \makeClusters \fragment
>>

```



## 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 " | "
```

```
}
```

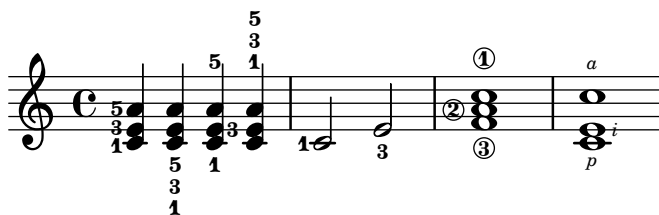


## Controlling the placement of chord fingerings

The placement of fingering numbers can be controlled precisely by using the property `fingeringOrientation`. For fingering orientation to apply, the fingering command must be used within a chord construct (`<...>`), even for single notes. Orientation for string numbers and right-hand fingerings may be controlled in a similar way by using the properties `stringNumberOrientation` and `strokeFingerOrientation`, respectively.

These properties can be set to a list of one to three values. They control whether fingerings may be placed above (if `up` appears in the list), below (if `down` appears), to the left (if `left` appears), or to the right (if `right` appears). Conversely, if a location is not listed, no fingering is placed there. LilyPond takes these constraints and works out the best placement for the fingering of the notes of the following chords. Note that `left` and `right` are mutually exclusive – fingerings may be placed only on one side or the other, not both.

```
\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
  \set stringNumberOrientations = #'(up left down)
  <f\3 a\2 c\1>1
  \set strokeFingerOrientations = #'(down right up)
  <c\rightHandFinger 1 e\rightHandFinger 2 c'\rightHandFinger 4 >
}
```



## Cow and ride bell example

Two different bells, entered with `'cb'` (cow bell) and `'rb'` (ride bell).

```
#(define mydrums '((ridebell default #f 3)
  (cowbell default #f -2)))
```

```

\new DrumStaff \with { instrumentName = #"Different Bells" }

\drummode {
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \set DrumStaff.clefPosition = 0.5
  \override DrumStaff.StaffSymbol.line-positions = #'(-2 3)
  \override Staff.BarLine.bar-extent = #'(-1.0 . 1.5)

  \time 2/4
  rb8 8 cb8 16 rb16-> ~ |
  16 8 16 cb8 8 |
}

```



## Creating blank staves

To create blank staves, generate empty measures then remove the `Bar_number_engraver` from the `Score` context, and the `Time_signature_engraver`, `Clef_engraver` and `Bar_engraver` from the `Staff` context.

```

#(set-global-staff-size 10) % for the documentation
% #(set-global-staff-size 20) % for letter and A4

```

```

\book {
  \score {
    { \repeat unfold 12 { s1 \break } }

    \layout {
      indent = 0
      \context {
        \Staff
        \remove "Time_signature_engraver"
        \remove "Clef_engraver"
        \remove "Bar_engraver"
      }
      \context {
        \Score
        \remove "Bar_number_engraver"
      }
    }
  }
}

```

```

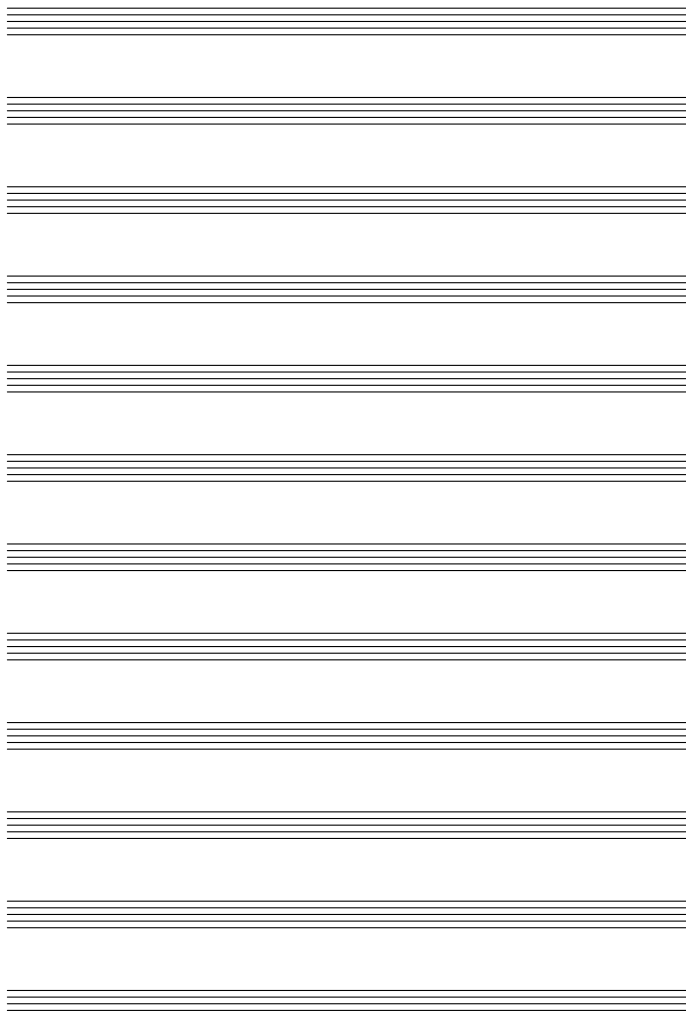
% for the documentation
\paper {
  #(set-paper-size "a6")
  ragged-last-bottom = ##f
  line-width = 90\mm
  left-margin = 7.5\mm
  bottom-margin = 5\mm
}

```

```
    top-margin = 5\mm
    tagline = ##f
}

% uncomment these lines for "letter" size
%{
\paper {
  #(set-paper-size "letter")
  ragged-last-bottom = ##f
  line-width = 7.5\in
  left-margin = 0.5\in
  bottom-margin = 0.25\in
  top-margin = 0.25\in
  tagline = ##f
}
}%

% uncomment these lines for "A4" size
%{
\paper {
  #(set-paper-size "a4")
  ragged-last-bottom = ##f
  line-width = 180\mm
  left-margin = 15\mm
  bottom-margin = 10\mm
  top-margin = 10\mm
  tagline = ##f
}
}%
}
```



## Custodes

Custodes may be engraved in various styles.

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

\markup \with-true-dimensions % work around a cropping issue
\score {
  \new Staff \with { \consists "Custos_engraver" } \relative c' {
    \override Staff.Custos.neutral-position = #4

    \override Staff.Custos.style = #'hufnagel
    c1^"hufnagel" \break
    <d a' f'>1

    \override Staff.Custos.style = #'medicaea
    c1^"medicaea" \break
    <d a' f'>1

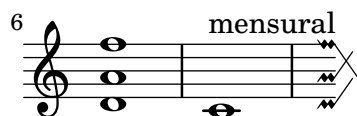
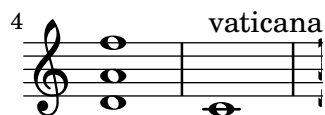
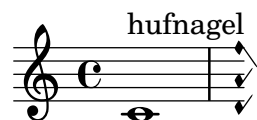
    \override Staff.Custos.style = #'vaticana
    c1^"vaticana" \break
```

```

<d a' f'>1

\override Staff.Custos.style = #'mensural
c1^"mensural" \break
<d a' f'>1
}
}

```



## Demo of MIDI instruments

Problem: How to know which `midiInstrument` values would be best for your composition?

Solution: A LilyPond demo file. You have to compile this snippet by yourself and listen to the created MIDI output file.

```

melody = \relative c' {
  \tempo 4 = 150
  c4.\mf g c16 b' c d
  e16 d e f g4 g'4 r
  R1
}

\score {
  \new Voice \melody
  \layout { }
}

\score {
  \new Voice {
    r\mf
    % 1-8 keyboard
    \set Staff.midiInstrument = "acoustic grand" \melody
    \set Staff.midiInstrument = "bright acoustic" \melody

```

```

\set Staff.midiInstrument = "electric grand" \melody
\set Staff.midiInstrument = "honky-tonk" \melody
\set Staff.midiInstrument = "electric piano 1" \melody
\set Staff.midiInstrument = "electric piano 2" \melody
\set Staff.midiInstrument = "harpsichord" \melody
\set Staff.midiInstrument = "clav" \melody

% 9-16 chrom percussion
\set Staff.midiInstrument = "celesta" \melody
\set Staff.midiInstrument = "glockenspiel" \melody
\set Staff.midiInstrument = "music box" \melody
\set Staff.midiInstrument = "vibraphone" \melody
\set Staff.midiInstrument = "marimba" \melody
\set Staff.midiInstrument = "xylophone" \melody
\set Staff.midiInstrument = "tubular bells" \melody
\set Staff.midiInstrument = "dulcimer" \melody

% 17-24 organ
\set Staff.midiInstrument = "drawbar organ" \melody
\set Staff.midiInstrument = "percussive organ" \melody
\set Staff.midiInstrument = "rock organ" \melody
\set Staff.midiInstrument = "church organ" \melody
\set Staff.midiInstrument = "reed organ" \melody
\set Staff.midiInstrument = "accordion" \melody
\set Staff.midiInstrument = "harmonica" \melody
\set Staff.midiInstrument = "concertina" \melody

% 25-32 guitar
\set Staff.midiInstrument = "acoustic guitar (nylon)" \melody
\set Staff.midiInstrument = "acoustic guitar (steel)" \melody
\set Staff.midiInstrument = "electric guitar (jazz)" \melody
\set Staff.midiInstrument = "electric guitar (clean)" \melody
\set Staff.midiInstrument = "electric guitar (muted)" \melody
\set Staff.midiInstrument = "overdriven guitar" \melody
\set Staff.midiInstrument = "distorted guitar" \melody
\set Staff.midiInstrument = "guitar harmonics" \melody

% 33-40 bass
\set Staff.midiInstrument = "acoustic bass" \melody
\set Staff.midiInstrument = "electric bass (finger)" \melody
\set Staff.midiInstrument = "electric bass (pick)" \melody
\set Staff.midiInstrument = "fretless bass" \melody
\set Staff.midiInstrument = "slap bass 1" \melody
\set Staff.midiInstrument = "slap bass 2" \melody
\set Staff.midiInstrument = "synth bass 1" \melody
\set Staff.midiInstrument = "synth bass 2" \melody

% 41-48 strings
\set Staff.midiInstrument = "violin" \melody
\set Staff.midiInstrument = "viola" \melody
\set Staff.midiInstrument = "cello" \melody
\set Staff.midiInstrument = "contrabass" \melody

```

```

\set Staff.midiInstrument = "tremolo strings" \melody
\set Staff.midiInstrument = "pizzicato strings" \melody
\set Staff.midiInstrument = "orchestral harp" \melody
\set Staff.midiInstrument = "timpani" \melody

% 49-56 ensemble
\set Staff.midiInstrument = "string ensemble 1" \melody
\set Staff.midiInstrument = "string ensemble 2" \melody
\set Staff.midiInstrument = "synthstrings 1" \melody
\set Staff.midiInstrument = "synthstrings 2" \melody
\set Staff.midiInstrument = "choir aahs" \melody
\set Staff.midiInstrument = "voice oohs" \melody
\set Staff.midiInstrument = "synth voice" \melody
\set Staff.midiInstrument = "orchestra hit" \melody

% 57-64 brass
\set Staff.midiInstrument = "trumpet" \melody
\set Staff.midiInstrument = "trombone" \melody
\set Staff.midiInstrument = "tuba" \melody
\set Staff.midiInstrument = "muted trumpet" \melody
\set Staff.midiInstrument = "french horn" \melody
\set Staff.midiInstrument = "brass section" \melody
\set Staff.midiInstrument = "synthbrass 1" \melody
\set Staff.midiInstrument = "synthbrass 2" \melody

% 65-72 reed
\set Staff.midiInstrument = "soprano sax" \melody
\set Staff.midiInstrument = "alto sax" \melody
\set Staff.midiInstrument = "tenor sax" \melody
\set Staff.midiInstrument = "baritone sax" \melody
\set Staff.midiInstrument = "oboe" \melody
\set Staff.midiInstrument = "english horn" \melody
\set Staff.midiInstrument = "bassoon" \melody
\set Staff.midiInstrument = "clarinet" \melody

% 73-80 pipe
\set Staff.midiInstrument = "piccolo" \melody
\set Staff.midiInstrument = "flute" \melody
\set Staff.midiInstrument = "recorder" \melody
\set Staff.midiInstrument = "pan flute" \melody
\set Staff.midiInstrument = "blown bottle" \melody
\set Staff.midiInstrument = "shakuhachi" \melody
\set Staff.midiInstrument = "whistle" \melody
\set Staff.midiInstrument = "ocarina" \melody

% 81-88 synth lead
\set Staff.midiInstrument = "lead 1 (square)" \melody
\set Staff.midiInstrument = "lead 2 (sawtooth)" \melody
\set Staff.midiInstrument = "lead 3 (calliope)" \melody
\set Staff.midiInstrument = "lead 4 (chiff)" \melody
\set Staff.midiInstrument = "lead 5 (charang)" \melody
\set Staff.midiInstrument = "lead 6 (voice)" \melody

```

```
\set Staff.midiInstrument = "lead 7 (fifths)" \melody
\set Staff.midiInstrument = "lead 8 (bass+lead)" \melody
```

```
% 89-96 synth pad
```

```
\set Staff.midiInstrument = "pad 1 (new age)" \melody
\set Staff.midiInstrument = "pad 2 (warm)" \melody
\set Staff.midiInstrument = "pad 3 (polysynth)" \melody
\set Staff.midiInstrument = "pad 4 (choir)" \melody
\set Staff.midiInstrument = "pad 5 (bowed)" \melody
\set Staff.midiInstrument = "pad 6 (metallic)" \melody
\set Staff.midiInstrument = "pad 7 (halo)" \melody
\set Staff.midiInstrument = "pad 8 (sweep)" \melody
```

```
% 97-104 synth effects
```

```
\set Staff.midiInstrument = "fx 1 (rain)" \melody
\set Staff.midiInstrument = "fx 2 (soundtrack)" \melody
\set Staff.midiInstrument = "fx 3 (crystal)" \melody
\set Staff.midiInstrument = "fx 4 (atmosphere)" \melody
\set Staff.midiInstrument = "fx 5 (brightness)" \melody
\set Staff.midiInstrument = "fx 6 (goblins)" \melody
\set Staff.midiInstrument = "fx 7 (echoes)" \melody
\set Staff.midiInstrument = "fx 8 (sci-fi)" \melody
```

```
% 105-112 ethnic
```

```
\set Staff.midiInstrument = "sitar" \melody
\set Staff.midiInstrument = "banjo" \melody
\set Staff.midiInstrument = "shamisen" \melody
\set Staff.midiInstrument = "koto" \melody
\set Staff.midiInstrument = "kalimba" \melody
\set Staff.midiInstrument = "bagpipe" \melody
\set Staff.midiInstrument = "fiddle" \melody
\set Staff.midiInstrument = "shanai" \melody
```

```
% 113-120 percussive
```

```
\set Staff.midiInstrument = "tinkle bell" \melody
\set Staff.midiInstrument = "agogo" \melody
\set Staff.midiInstrument = "steel drums" \melody
\set Staff.midiInstrument = "woodblock" \melody
\set Staff.midiInstrument = "taiko drum" \melody
\set Staff.midiInstrument = "melodic tom" \melody
\set Staff.midiInstrument = "synth drum" \melody
\set Staff.midiInstrument = "reverse cymbal" \melody
```

```
% 121-128 sound effects
```

```
\set Staff.midiInstrument = "guitar fret noise" \melody
\set Staff.midiInstrument = "breath noise" \melody
\set Staff.midiInstrument = "seashore" \melody
\set Staff.midiInstrument = "bird tweet" \melody
\set Staff.midiInstrument = "telephone ring" \melody
\set Staff.midiInstrument = "helicopter" \melody
\set Staff.midiInstrument = "applause" \melody
\set Staff.midiInstrument = "gunshot" \melody
```



```

}
\midi { }
}

```



## Direction of merged ‘fa’ shape note heads

Using property `NoteCollision.fa-merge-direction`, the direction of “fa” shape note heads (“fa”, “faThin”, etc.) can be controlled independently of the stem direction if two voices with the same pitch and different stem directions are merged. If this property is not set, the “down” glyph variant is used.

```

{
  \clef bass

  << { \aikenHeads
    f2
    \override Staff.NoteCollision.fa-merge-direction = #UP
    f2 }
  \\ { \aikenHeads
    f2
    f2 }
  >>
}

```



## Embedding native PostScript in a \markup block

PostScript code can be directly inserted inside a `\markup` block.

In general it is recommended to use LilyPond’s native graphical markup commands like `\polygon` instead, which can be used with all LilyPond backends.

```

\relative c' {
  a2-\markup \postscript "0 3 moveto
                        5 2 rlineto
                        stroke"
  -\markup \postscript "[1 1] 0 setdash
                        0 0 moveto
                        5 2 rlineto
                        stroke"
  b2-\markup \postscript "1 1 moveto
                        0 0 1 2 8 4 10 2 rcurveto
                        stroke"
  a'1
}

```



## Engravers one by one

LilyPond handles the various elements necessary to typeset a score with plugins. Each plugin is called an *engraver*. In this example, (some) engravers are switched on one by one, in the following order:

- note heads,
- staff symbol,
- clef,
- stem,
- beams, slurs, accents,
- accidentals, bar lines, time signature, and key signature.

Engravers are grouped. For example, note heads, slurs, beams, etc., form a *Voice* context. Engravers for key signature, accidentals, bar line, etc., form a *Staff* context.

```

topVoice = \relative c' {
  \key d \major
  es8([ g] a[ fis])
  b4
  b16[-. b-. b-. cis-.]
  d4->
}

% empty staff and voice contexts
MyStaff = \context {
  \type Engraver_group
  \name Staff
  \accepts Voice
  \defaultchild Voice
}

MyVoice = \context {
  \type Engraver_group
  \name Voice
}

% add note heads
MyVoice = \context {
  \MyVoice
  \consists Note_heads_engraver
}

\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

```

```

% add staff
MyStaff = \context {
  \MyStaff
  \consists Staff_symbol_engraver
}
\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

% add clef
MyStaff = \context {
  \MyStaff
  \consists Clef_engraver
}
\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

% add stems
MyVoice = \context {
  \MyVoice
  \consists Stem_engraver
}
\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

% add beams, slurs, and accents
MyVoice = \context {
  \MyVoice
  \consists Beam_engraver
  \consists Slur_engraver
  \consists Script_engraver
  \consists Rhythmic_column_engraver
}
\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

```

```

    }
}

% add accidentals, bar, time signature, and key signature
MyStaff = \context {
  \MyStaff
  \consists Accidental_engraver
  \consists Bar_engraver
  \consists Time_signature_engraver
  \consists Key_engraver
}
\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

```



## Flamenco notation

For flamenco guitar, some special notation is used.

- A *golpe* symbol indicates a slap on the guitar body with the nail of the ring finger.
- An arrow indicates (the direction of) strokes.
- Different letters for fingering are used (“p”: thumb, “i”: index finger, “m”: middle finger, “a”: ring finger and “x”: little finger).
- Marking 3- and 4-finger *rasgueados*: stroke upwards with all fingers, ending with an up-and down using the index finger.
- *Abanicos* are strokes (in tuples) with thumb (down), little and index finger (both up). There’s also an *abanico 2* where middle and ring finger are used instead of the little finger.
- *Alza pua* indicates fast playing with the thumb.

Most figures use arrows in combination with fingering; with abanicos and rasgueados, note heads are printed only for the first chord.

This snippet contains some header-like code that can be copied as `flamenco.ly` and included in source files.

```
%%%%%%%% Cut here ----- Start of `flamenco.ly`.

% Text indicators.
abanico = ^\markup \small { \italic Abanico }
rasgueado = ^\markup \small { \italic Ras. }
alzapua = ^\markup \small { \italic Alzapua }

% Finger stroke symbols.
strokeUp = \markup {
  \combine
    \override #'(thickness . 1.3) \draw-line #'(0 . 2)
    \raise #2 \arrow-head #Y #UP ##f }
strokeDown = \markup {
  \combine
    \arrow-head #Y #DOWN ##f
    \override #'(thickness . 1.3) \draw-line #'(0 . 2) }

% Golpe symbol.
golpe = \markup {
  \filled-box #'(0 . 1) #'(0 . 1) #0
  \hspace #-1.6
  \with-color #white
  \filled-box #'(0.15 . 0.85) #'(0.15 . 0.85) #0
}

% Strokes, fingers, and golpe command.
RHp = \rightHandFinger #1
RH2 = \rightHandFinger #2
RHm = \rightHandFinger #3
RHa = \rightHandFinger #4
RHx = \rightHandFinger #5
RHu = \rightHandFinger \strokeUp
RHd = \rightHandFinger \strokeDown
RHg = \rightHandFinger \golpe

% Various shorthands.
tupletOff = {
  \once \omit TupletNumber
  \once \omit TupletBracket
}

tupletsOff = {
  \omit TupletNumber
  \override TupletBracket.bracket-visibility = #'if-no-beam
}

tupletsOn = {
```

```

\override TupletBracket.bracket-visibility = #'default
\undo \omit TupletNumber
}

```

```

headsOff = {
  \hide TabNoteHead
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}

```

```

headsOn = {
  \override TabNoteHead.transparent = ##f
  \override NoteHead.transparent = ##f
  \override NoteHead.no-ledgers = ##f
}

```

```

%%%%%%%% Cut here ----- End of `flamenco.ly`.

```

```

part = \relative c' {
  \set strokeFingerOrientations = #'(up)
  \key a\major

```

```

  <a, e' a cis e\RHu\RHl>8
    <a e' a cis e\RHd\RHl>8
    r4
    r2~\markup\golpe |
  <a e' a cis e\RHu\RHl>8
    <a e' a cis e\RHd\RHl>8
    <a e' a cis e\RHu\RHl\RHg>8
    <a e' a cis e\RHd\RHl>8
    r2 |
  <a e' a cis e\RHu\RHl>16\rasgueado
    \headsOff
    <a e' a cis e\RHu\RHm>
    <a e' a cis e\RHu\RHl>
    <a e' a cis e\RHd\RHl>~
    \headsOn
    <a e' a cis e>2
    r4 |
  \tupletOff
  \tuplet 5/4 {
    <a e' a cis e\RHu\RHx>16\rasgueado
    \headsOff
    <a e' a cis e\RHu\RHl>
    <a e' a cis e\RHu\RHm>
    <a e' a cis e\RHu\RHl>
    <a e' a cis e\RHd\RHl>~
    \headsOn
  }
  <a e' a cis e>2
  r4 |

```

Treble:  $\text{G} \text{ A} \text{ B}$   
 Alto:  $\text{0} \text{ 0}$   
 Bass:  $\text{0} \text{ 0}$

The image shows a musical score for two pieces, 'Abanico' and 'Alzapua'. The score is written on a grand staff with a treble clef and a key signature of two sharps (F# and C#). The time signature is 2/4. The 'Abanico' section consists of four measures of music, each with a specific fingering or articulation indicated by letters (p, x, i) and arrows (up and down). The 'Alzapua' section consists of four measures of music, each with a specific fingering or articulation indicated by letters (p, x, i) and arrows (up and down). The score is written on a grand staff with a treble clef and a key signature of two sharps (F# and C#). The time signature is 2/4. The 'Abanico' section consists of four measures of music, each with a specific fingering or articulation indicated by letters (p, x, i) and arrows (up and down). The 'Alzapua' section consists of four measures of music, each with a specific fingering or articulation indicated by letters (p, x, i) and arrows (up and down).

## High and low woodblock example

Two Woodblocks, entered with 'wbh' (high woodblock) and 'wbl' (low woodblock). The length of the bar line has been altered with an `\override` command, otherwise it would be too short. The positions of the two staff lines also have to be explicitly defined.

```
% 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 #f 3)
                  (lowwoodblock default #f -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 = #'(-1.0 . 1.5)
  % small correction for the clef:
  \set DrumStaff.clefPosition = 0.5
}
```

```
\new DrumStaff {
  % with this you load your new drum style table
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)

  \woodstaff

  \drummode {
    \time 2/4
    wbh8 16 16 8-> 8 |
    wbl8 16 16-> ~ 16 16 r8 |
  }
}
```

The image shows a musical notation for a drum staff. It is written on a single staff with a 2/4 time signature. The notation includes eighth notes, sixteenth notes, and rests, with some notes having accents or other markings above them.



## How to change fret diagram position

If you want to move the position of a fret diagram, for example, to avoid collision, or to place it between two notes, you have various possibilities.

- 1) Modify the value of the padding or extra-offset property (as shown in the first line).
- 2) You can add an invisible voice and attach the fret diagrams to the invisible notes in that voice (as shown in the second line).

If you need to move the fret according with a rhythmic position inside the bar (in the example, the third beat of the measure) the second example is better, because the fret is aligned with the third beat itself.

```

harmonies = \chordmode
{
  a8:13
  \once \override ChordNames.ChordName.extra-offset = #'(10 . 0)
  b8:13 s4. |
  s2 b2:13
}

\score {
  <<
    \new ChordNames \harmonies
    \new Staff {
      % Method 1.
      a8~\markup \fret-diagram "6-x;5-0;4-2;3-0;2-0;1-2;"
      \once \override TextScript.extra-offset = #'(10 . 0)
      b4.~\markup \fret-diagram "6-x;5-2;4-4;3-2;2-2;1-4;"
      b4. a8 | \break

      % Method 2.
      <<
        { a8 b4.~ b4. a8 }
        { s2 s2~\markup \fret-diagram "6-x;5-2;4-4;3-2;2-2;1-4;" }
      >> |
    }
  >>
}

```

## How to put ties between syllables in lyrics

This can be achieved by separating those syllables by tildes.

```
\lyrics {
  wa~o~a
}

wa_o_a
```

## Laissez vibrer ties

Laissez vibrer ties have a fixed size. Their positioning can be tuned using the `tie-configuration` property.

See also snippet “Longer laissez vibrer ties”.

```
\relative c' {
  <c e g>4\laissezVibrer r <c f g>\laissezVibrer r
  <c d f g>4\laissezVibrer r <c d f g>4.\laissezVibrer r8

  <c d e f>4\laissezVibrer r
  \override LaissezVibrerTieColumn.tie-configuration
    = #`((-7 . ,DOWN)
        (-5 . ,DOWN)
        (-3 . ,UP)
        (-1 . ,UP))
  <c d e f>4\laissezVibrer r
}
```



## Percussion example

A short example taken from Stravinsky's *L'histoire du Soldat*.

```
#(define mydrums '((bassdrum default #f 4)
                   (snare default #f -4)
                   (tambourine default #f 0)))
```

```
U = \stemUp
D = \stemDown
```

```
global = {
  \time 3/8 s4.
  \time 2/4 s2*2
  \time 3/8 s4.
  \time 2/4 s2
}
```

```
drumsA = {
  \context DrumVoice <<
    \global
    \drummode {
      \autoBeamOff
```

```

\D sn8 \U tamb s |
sn4 \D sn4 |
\U tamb8 \D sn \U sn16 \D sn \U sn8 |
\D sn8 \U tamb s |
\U sn4 s8 \U tamb
}
>>
}

drumsB = \drummode {
  s4 bd8 s2*2 s4 bd8 s4 bd8 s
}

\layout {
  indent = 40\mm
  \context {
    \DrumStaff
    drumStyleTable = #(alist->hash-table mydrums)
  }
}

\score {
  \new StaffGroup <<
    \new DrumStaff \with {
      instrumentName = \markup \center-column {
        "Tambourine"
        "et"
        "caisse claire s. timbre" }
    } \drumsA
    \new DrumStaff \with {
      instrumentName = "Grosse Caisse"
    } \drumsB
  >>
}

```

Tambourine  
et  
caisse claire s. timbre

Grosse Caisse

## Tam-tam example

A tam-tam example, entered with 'tt'.

```
#(define mydrums '((tamtam default #f 0)))
```

```

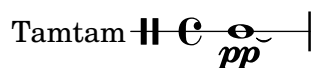
\new DrumStaff \with { instrumentName = #"Tamtam" }

\drummode {
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \override Staff.StaffSymbol.line-positions = #'( 0 )
}

```

```
\override Staff.BarLine.bar-extent = #'(-1.5 . 1.5)

tt 1 \pp \laissezVibrer
}
```



## Tambourine example

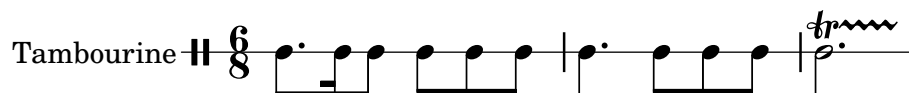
A tambourine example, entered with ‘tamb’.

```
#(define mydrums '((tambourine default #f 0)))

\new DrumStaff \with { instrumentName = #"Tambourine" }

\drummode {
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \override Staff.StaffSymbol.line-positions = #'( 0 )
  \override Staff.BarLine.bar-extent = #'(-1.5 . 1.5)

  \time 6/8
  tamb8. 16 8 8 8 8 |
  tamb4. 8 8 8 |
  % The trick with the scaled duration and the shorter rest
  % is necessary for the correct ending of the trill-span!
  tamb2.*5/6 \startTrillSpan s8 \stopTrillSpan |
}
```



## Time signature in brackets

The time signature can be enclosed within brackets.

```
\relative c' {
  \override Staff.TimeSignature.stencil = #(lambda (grob)
    (bracketify-stencil (ly:time-signature::print grob) Y 0.1 0.2 0.1))
  \time 2/4
  a4 b8 c
}
```



## Time signature in parentheses

The time signature can be enclosed within parentheses.

```
\relative c' {
  \override Staff.TimeSignature.stencil = #(lambda (grob)
    (parenthesize-stencil (ly:time-signature::print grob) 0.1 0.4 0.4 0.1))
  \time 2/4
  a4 b8 c
}
```

}



## Using an extra voice for breaks

Often it is easier to manage line and page-breaking information by keeping it separate from the music by introducing an extra voice containing only skips along with the `\break`, `\pageBreak`, and other layout information.

This pattern becomes especially helpful when overriding `line-break-system-details` and the other useful but long properties of the `NonMusicalPaperColumn` grob.

```
music = \relative c'' { c4 c c c }
```

```
\score {
  \new Staff <<
    \new Voice {
      s1*2 \break
      s1*3 \break
      s1*4 \break
      s1*5 \break
    }
    \new Voice {
      \repeat unfold 2 { \music }
      \repeat unfold 3 { \music }
      \repeat unfold 4 { \music }
      \repeat unfold 5 { \music }
    }
  >>
}
```

```
\paper {
  indent = 0
  line-width = 140\mm
  ragged-right = ##t
}
```



## 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 #'() }
  c1^\markup \center-column { "piccolo"
    " "
    \woodwind-diagram #'piccolo #'() }
  c1^\markup \center-column { "flute"
    " "
    \woodwind-diagram #'flute #'() }
  c1^\markup \center-column { "oboe"
    " "
    \woodwind-diagram #'oboe #'() }
  c1^\markup \center-column { "clarinet"
    " "
    \woodwind-diagram #'clarinet #'() }

  \break

  c1^\markup \center-column { "bass clarinet"
    " "
    \woodwind-diagram #'bass-clarinet #'() }
  c1^\markup \center-column { "saxophone"
    " "
    \woodwind-diagram #'saxophone #'() }
  c1^\markup \center-column { "bassoon"
    " "
    \woodwind-diagram #'bassoon #'() }
  c1^\markup \center-column { "contrabassoon"
    " "
    \woodwind-diagram #'contrabassoon #'() }
}

\paper {
  \system-system-spacing.padding = 5
}
```

The image displays musical notation for nine woodwind instruments, arranged in two rows. Each instrument is represented by a specific set of notes and rests on a five-line staff, with a treble clef and a common time signature (C) at the beginning of each row. The instruments are labeled above their respective staves.

**Top Row:**

- tin whistle:** Notes are placed on the first, second, and third lines of the staff.
- piccolo:** Notes are placed on the first, second, and third lines of the staff.
- flute:** Notes are placed on the first, second, and third lines of the staff.
- oboe:** Notes are placed on the first, second, and third lines of the staff.
- clarinet:** Notes are placed on the first, second, and third lines of the staff.

**Bottom Row:**

- bass clarinet:** Notes are placed on the first, second, and third lines of the staff.
- saxophone:** Notes are placed on the first, second, and third lines of the staff.
- bassoon:** Notes are placed on the first, second, and third lines of the staff.
- contrabassoon:** Notes are placed on the first, second, and third lines of the staff.

The notation for each instrument is unique, reflecting its specific range and characteristics. The notes are represented by circles with stems, and the rests are represented by horizontal lines.

## 35 Symbols and glyphs

### Accordion register symbols

Accordion register symbols are available as `\markup` as well as as standalone music events (as register changes tend to occur between actual music events). Bass registers are not overly standardized. The available commands can be found in ‘Discant symbols’ in the Notation Reference (<https://lilypond.org/doc/v2.24/Documentation/notation/accordion#discant-symbols>).

```
#(use-modules (lily 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" }
    >>
  }
>>
```

The image shows a musical score for piano and accordion. The piano part is written on two staves (treble and bass). The accordion part is written on a single staff with a treble clef. The score includes various musical notations such as notes, rests, and chords. The accordion part uses discant symbols (b, am, gm, a) to indicate register changes. The piano part includes a bass line with notes D, C, B, and A.



## Adding indicators to staves which get split after a break

This snippet defines the commands `\splitStaffBarLine`, `\convUpStaffBarLine`, and `\convDownStaffBarLine`. These add arrows at a bar line to denote that several voices sharing a staff will each continue on a staff of their own in the next system, or that voices split in this way recombine.

Note that the implementation in this snippet draws dimensionless arrows into the right margin. For normal printing, this doesn't cause problems. However, it is necessary to increase the bounding box horizontally if you render the code as an image to avoid cropping, as demonstrated below.

```
#(define-markup-command (arrow-at-angle layout props angle-deg length fill)
  (number? number? boolean?)
  (let* ((PI-OVER-180 (/ (atan 1 1) 34))
        (degrees->radians (lambda (degrees) (* degrees PI-OVER-180)))
        (angle-rad (degrees->radians angle-deg))
        (target-x (* length (cos angle-rad)))
        (target-y (* length (sin angle-rad))))
    (interpret-markup layout props
      (markup
        #:translate (cons (/ target-x 2) (/ target-y 2))
        #:rotate angle-deg
        #:translate (cons (/ length -2) 0)
        #:concat (#:draw-line (cons length 0)
          #:arrow-head X RIGHT fill))))))

splitStaffBarLineMarkup = \markup \with-dimensions #'(0 . 0) #'(0 . 0) {
  \combine
  \arrow-at-angle #45 #(sqrt 8) ##t
  \arrow-at-angle #-45 #(sqrt 8) ##t
}

splitStaffBarLine = {
  \once \override Staff.BarLine.stencil =
  #(lambda (grob)
    (ly:stencil-combine-at-edge
      (ly:bar-line::print grob)
      X RIGHT
      (grob-interpret-markup grob splitStaffBarLineMarkup)
      0))
  \break
}

convDownStaffBarLine = {
  \once \override Staff.BarLine.stencil =
  #(lambda (grob)
    (ly:stencil-combine-at-edge
      (ly:bar-line::print grob)
      X RIGHT
      (grob-interpret-markup grob #{
        \markup\with-dimensions #'(0 . 0) #'(0 . 0) {
          \translate #'(0 . -.13)\arrow-at-angle #-45 #(sqrt 8) ##t
        }#}))
  }
```

```

    0))
  \break
}

convUpStaffBarLine = {
  \once \override Staff.BarLine.stencil =
  #(\lambda (grob)
    (ly:stencil-combine-at-edge
      (ly:bar-line::print grob)
      X RIGHT
      (grob-interpret-markup grob #{
        \markup\with-dimensions #'(0 . 0) #'(0 . 0) {
          \translate #'(0 . .14)\arrow-at-angle #45 #(\sqrt 8) ##t
        }#})
    0))
  \break
}

\paper {
  indent = 10\mm
  short-indent = 10\mm
  line-width = 8\cm
}

separateSopranos = {
  \set Staff.instrumentName = "AI AII"
  \set Staff.shortInstrumentName = "AI AII"
  \splitStaffBarLine
  \change Staff = "up"
}

convSopranos = {
  \convDownStaffBarLine
  \change Staff = "shared"
  \set Staff.instrumentName = "S A"
  \set Staff.shortInstrumentName = "S A"
}

sI = {
  \voiceOne
  \repeat unfold 4 f''2
  \separateSopranos
  \repeat unfold 4 g''2
  \convSopranos
  \repeat unfold 4 c''2
}

sII = {
  s1*2
  \voiceTwo
  \change Staff = "up"
  \repeat unfold 4 d''2
}

aI = {

```

```

\voiceTwo
\repeat unfold 4 a'2
\voiceOne
\repeat unfold 4 b'2
\convUpStaffBarLine
\voiceTwo
\repeat unfold 4 g'2
}
aII = {
  s1*2
  \voiceTwo
  \repeat unfold 4 g'2
}
ten = {
  \voiceOne
  \repeat unfold 4 c'2
  \repeat unfold 4 d'2
  \repeat unfold 4 c'2
}
bas = {
  \voiceTwo
  \repeat unfold 4 f2
  \repeat unfold 4 g2
  \repeat unfold 4 c2
}

\markup \pad-x #3 % avoid cropping
\score {
  <<
  \new ChoirStaff <<
  \new Staff = up \with {
    instrumentName = "SI SII"
    shortInstrumentName = "SI SII"
  } {
    s1*4
  }

  \new Staff = shared \with {
    instrumentName = "S A"
    shortInstrumentName = "S A"
  } <<
  \new Voice = sopI \sI
  \new Voice = sopII \sII
  \new Voice = altI \aI
  \new Voice = altII \aII
  >>
  \new Lyrics \with {
    alignBelowContext = up
  }
  \lyricsto sopII { e f g h }
  \new Lyrics \lyricsto altI { a b c d e f g h i j k l }

```

```
\new Staff = men \with {  
  instrumentName = "T B"  
  shortInstrumentName = "T B"  
} <<  
  \clef F  
  \new Voice = ten \ten  
  \new Voice = bas \bas  
>>  
\new Lyrics \lyricsto bas { a b c d e f g h i j k l }  
>>  
>>  
  
\layout {  
  \context {  
    \Staff \RemoveEmptyStaves  
    \override VerticalAxisGroup.remove-first = ##t  
  }  
}  
}
```

The image displays three systems of musical notation, each with a Soprano (S A) and Tenor (T B) part. The first system shows notes 'a', 'b', 'c', 'd' on a four-line staff. The second system shows notes 'e', 'f', 'g', 'h' on a four-line staff, with a third part (T B) also showing 'e', 'f', 'g', 'h'. The third system shows notes 'i', 'j', 'k', 'l' on a four-line staff. The notation includes various clefs and note heads, some with flags or beams.

## Ancient fonts

This snippets shows many of the symbols contained in the Emmentaler font that are used by LilyPond for typesetting ancient notation.

```
m = { c1 e f ges cis' \bar "||" }
```

```
\markup \with-true-dimensions % work around a cropping issue
\score {
  \new VaticanaVoice {
    \clef "vaticana-fa2"
    \key es \major
    \textMark \markup \rounded-box "Vaticana clefs, custos and note heads"

    \override NoteHead.style = #'vaticana.punctum
    <>~"vaticana.punctum" \m

    \override NoteHead.style = #'vaticana.inclinatum
```

```

<>^"vaticana.inclinatum" \m

\override NoteHead.style = #'vaticana.quilisma
<>^"vaticana.quilisma" \m

\clef "vaticana-fa1"
\override NoteHead.style = #'vaticana.plica
<>^"vaticana.plica" \m

\override NoteHead.style = #'vaticana.reverse.plica
<>^"vaticana.reverse.plica" \m

\override NoteHead.style = #'vaticana.punctum.cavum
<>^"vaticana.punctum.cavum" \m

\override NoteHead.style = #'vaticana.lpes
<>^"vaticana.punctum.lpes" \m

\override NoteHead.style = #'vaticana.upes
<>^"vaticana.punctum.upes" \m

\override NoteHead.style = #'vaticana.vupes
<>^"vaticana.punctum.vupes" \m

\override NoteHead.style = #'vaticana.linea.punctum
<>^"vaticana.punctum.linea" \m

\override NoteHead.style = #'vaticana.epiphonus
<>^"vaticana.punctum.epiphonus" \m

\override NoteHead.style = #'vaticana.cephalicus
<>^"vaticana.punctum.cephalicus" \m

\break

\textMark \markup \rounded-box "Medicaea clefs, custos and note heads"
\set VaticanaStaff.alterationGlyphs =
  #alteration-medicaea-glyph-name-alist
\override VaticanaStaff.Custos.style = #'medicaea

\clef "medicaea-fa2"
\override NoteHead.style = #'medicaea.punctum
<>^"medicaea.punctum" \m

\clef "medicaea-do2"
\override NoteHead.style = #'medicaea.inclinatum
<>^"medicaea.inclinatum" \m

\override NoteHead.style = #'medicaea.virga
<>^"medicaea.virga" \m

\clef "medicaea-fa1"

```

```

\override NoteHead.style = #'medicaea.rvirga
<>^"medicaea.rvirga" \m

\break

\textMark \markup \rounded-box "Hufnagel clefs, custos and note heads"
\set Staff.alterationGlyphs =
  #alteration-hufnagel-glyph-name-alist
\override VaticanaStaff.Custos.style = #'hufnagel
\clef "hufnagel-fa2"

\break

\override NoteHead.style = #'hufnagel.punctum
<>^"hufnagel.punctum" \m

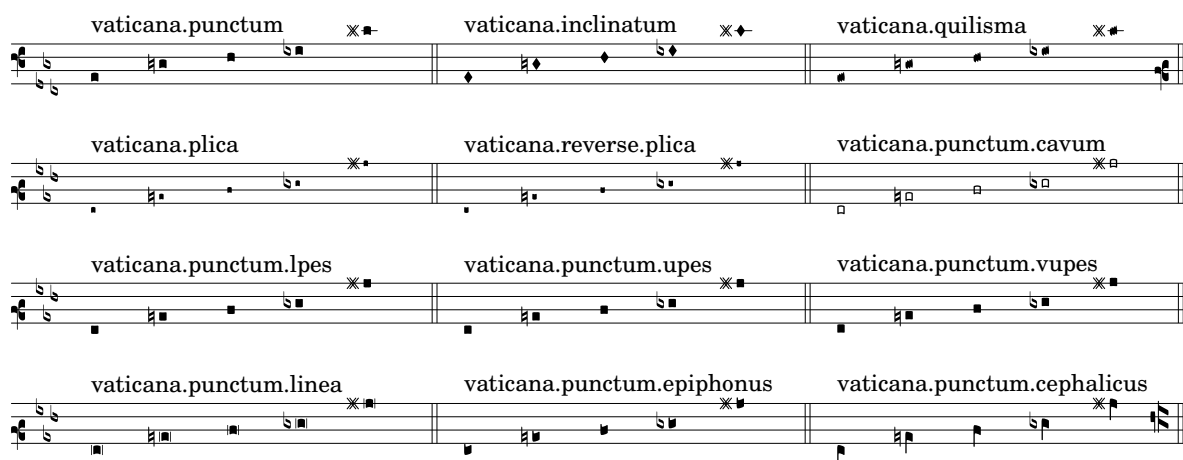
\clef "hufnagel-do2"
\override NoteHead.style = #'hufnagel.lpes
<>^"hufnagel.lpes" \m

\clef "hufnagel-do-fa"
\override NoteHead.style = #'hufnagel.virga
<>^"hufnagel.virga" \m
}

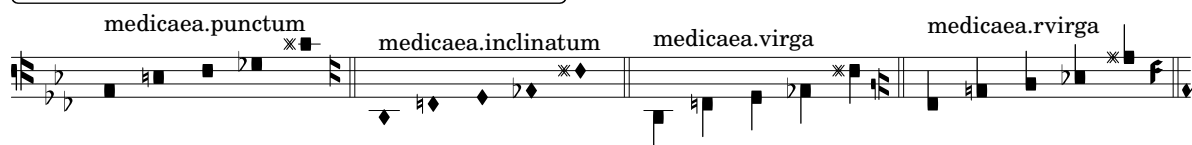
\layout {
  \context {
    \Score
    \override TextScript.font-size = #-2
    \override TextMark.break-align-symbols = #'(left-edge clef staff-bar)
    \override TextMark.padding = 4
    \omit BarNumber
  }
  \context {
    \VaticanaStaff
    alterationGlyphs =
      #alteration-vaticana-glyph-name-alist
  }
}
}

```

### Vaticana clefs, custos and note heads



### Medicaea clefs, custos and note heads



### Hufnagel clefs, custos and note heads



## Breathing signs

Breathing signs are available in different tastes: commas (default), ticks, vees and “railroad tracks” (caesura).

```
\new Staff \relative c'' {
  \key es \major
  \time 3/4
  % this bar contains no \breathe
  << { g4 as g } \ { es4 bes es } >> |
  % Modern notation:
  % by default, \breathe uses the rcomma, just as if saying:
  % \override BreathingSign.text =
  % #(make-musicglyph-markup "scripts.rcomma")
  << { g4 as g } \ { es4 \breathe bes es } >> |

  % rvarcomma and lvarcomma are variations of the default rcomma
  % and lcomma
  % N.B.: must use Staff context here, since we start a Voice below
  \override Staff.BreathingSign.text =
    \markup { \musicglyph "scripts.rvarcomma" }
  << { g4 as g } \ { es4 \breathe bes es } >> |

  % raltcomma and laltcomma are alternative variations of the
```



```

% default rcomma and lcomma
\override Staff.BreathingSign.text =
  \markup { \musicglyph "scripts.raltcomma" }
<< { g4 as g } \ { es4 \breathe bes es } >> |

% vee
\override BreathingSign.text =
  \markup { \musicglyph "scripts.uupbow" }
es8[ d es f g] \breathe f |

% caesura
\override BreathingSign.text =
  \markup { \musicglyph "scripts.caesura.curved" }
es8[ d] \breathe es[ f g f] |
es2 r4 \bar "||"
}

```



## Broken crescendo hairpin

In order to make parts of a crescendo hairpin invisible, the following method is used: A white rectangle is drawn on top of the respective part of the crescendo hairpin, making it invisible. The rectangle is defined as a text markup.

The markup command with-dimensions tells LilyPond to consider only the bottom edge of the rectangle when spacing it against the hairpin. The property staff-padding prevents the rectangle from fitting between the hairpin and staff.

Make sure the hairpin is in a lower layer than the text markup to draw the rectangle over the hairpin.

```

\relative c' {
  <<
    {
      \dynamicUp
      r2 r16 c'8.\pp r4
    }
    \
    {
      \override DynamicLineSpanner.layer = #0
      des,2\mf\< ~
      \override TextScript.layer = #2
      \once\override TextScript.staff-padding = #6
      \once\override TextScript.vertical-skylines = #'()
      des16_\markup \with-dimensions #'(2 . 7) #'(0 . 0)
        \with-color #white
        \filled-box #'(2 . 7) #'(0 . 2) #0
      r8. des4 ~ des16->\sff r8.
    }
  >>
}

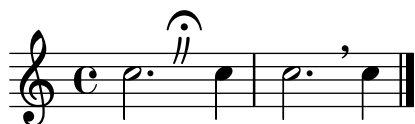
```



## Caesura (“railtracks”) with fermata

A caesura is sometimes denoted by a double “railtracks” breath mark with a fermata sign positioned above. This snippet shows an optically pleasing combination of railtracks and fermata.

```
\relative c' {
  c2.
  % construct the symbol
  \override BreathingSign.text = \markup {
    \override #'(direction . 1)
    \override #'(baseline-skip . 1.8)
    \dir-column {
      \translate #'(0.155 . 0)
      \center-align \musicglyph "scripts.caesura.curved"
      \center-align \musicglyph "scripts.ufermata"
    }
  }
  \breathe c4
  % set the breath mark back to normal
  \revert BreathingSign.text
  c2. \breathe c4
  \bar "|."
}
```



## Custodes

Custodes may be engraved in various styles.

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

\markup \with-true-dimensions % work around a cropping issue
\score {
  \new Staff \with { \consists "Custos_engraver" } \relative c' {
    \override Staff.Custos.neutral-position = #4

    \override Staff.Custos.style = #'hufnagel
    c1^"hufnagel" \break
    <d a' f'>1

    \override Staff.Custos.style = #'medicaea
    c1^"medicaea" \break
  }
}
```

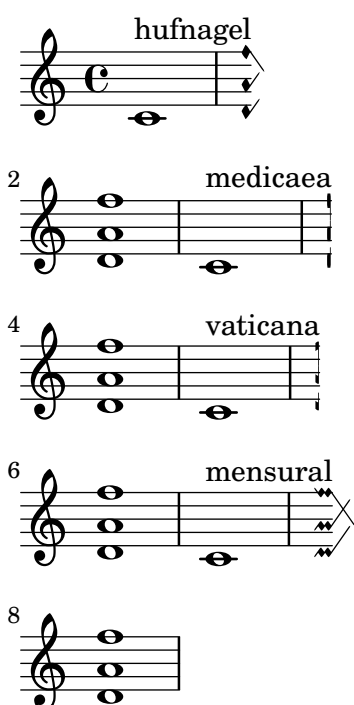
```

< d a' f' > 1

\override Staff.Custos.style = #'vaticana
c1^"vaticana" \break
< d a' f' > 1

\override Staff.Custos.style = #'mensural
c1^"mensural" \break
< d a' f' > 1
}
}

```



## Customizing the position and number of dots in repeat sign bar lines

If you want to customize the position and/or number of dots in repeat sign bar lines, you can define new custom bar lines or redefine the way default repeat signs are drawn. This snippet shows how. This may be particularly helpful when using a staff with custom line-positions, as shown in this snippet.

```

#(define ((make-custom-dot-bar-line dot-positions) is-span grob extent)

```

```

  "Draw dots (repeat sign dots) at @var{dot-positions}. The
  coordinates of @var{dot-positions} are equivalent to the
  coordinates of @code{StaffSymbol.line-positions}, a dot-position
  of X and a line-position of X indicate the same vertical position."

```

```

  (let* ((staff-space (ly:staff-symbol-staff-space grob))
        (dot (ly:font-get-glyph (ly:grob-default-font grob) "dots.dot"))
        (stencil empty-stencil))
    (for-each
      (lambda (dp)

```

```

      (set! stencil (ly:stencil-add stencil
                                   (ly:stencil-translate-axis dot (* dp (/ staff-space 2)) Y)))
      dot-positions)
      stencil))

% With the procedure above we can define custom bar-lines, for example,
% one that resembles standard repeat sign bar lines except it has
% three dots at staff positions -3, 0, and 3.

#(add-bar-glyph-print-procedure "*" (make-custom-dot-bar-line '(-3 0 3)))
\defineBarLine ".|*" #'(" " " " " ")
\defineBarLine "*|." #'(" " " " " ")

% We can also customize the dot positions used in all default repeat signs
% by redefining the print procedure of the colon bar glyph (":"). On a staff
% with line-positions of '(-4 -2 2 4) the default repeat sign dots appear
% at '(-3 3), but we can put them at '(-1 1) instead.

#(add-bar-glyph-print-procedure ":" (make-custom-dot-bar-line '(-1 1)))

\new Staff \with {
  \override StaffSymbol.line-positions = #'(-4 -2 2 4)
  \override StaffSymbol.staff-space = #1.3
} {
  \relative f' {
    g1
    \bar ".|*"
    g
    \bar "*|."
    g
    \bar ".|:-|"
    g
    \bar ":|. "
    g
    \repeat volta 2 {
      g
    }
  }
}

```



## Fingering symbols for wind instruments

Special symbols can be achieved by combining existing glyphs, which is useful for wind instruments.

```

lineup =
  \tweak outside-staff-padding #0
  \tweak staff-padding #0

```

```

\ tweak padding #0.2
\ tweak parent-alignment-X #CENTER
\ tweak self-alignment-X #CENTER
\ etc

\relative c' {
  g\open
  g\lineup ~\markup \combine
    \musicglyph "scripts.open"
    \musicglyph "scripts.tenuto"
  g\lineup ~\markup \combine
    \musicglyph "scripts.open"
    \musicglyph "scripts.stopped"
  g\stopped
}

```



## How to put ties between syllables in lyrics

This can be achieved by separating those syllables by tildes.

```

\lyrics {
  wa~o~a
}

wa_o_a

```

## Positioning segno and coda (with line break)

If you want to place an exiting segno sign and add text like “D.S. al Coda” next to it where usually the staff lines are you can use this snippet. The coda will resume in a new line. There is a variation documented in this snippet, where the coda will remain on the same line.

```

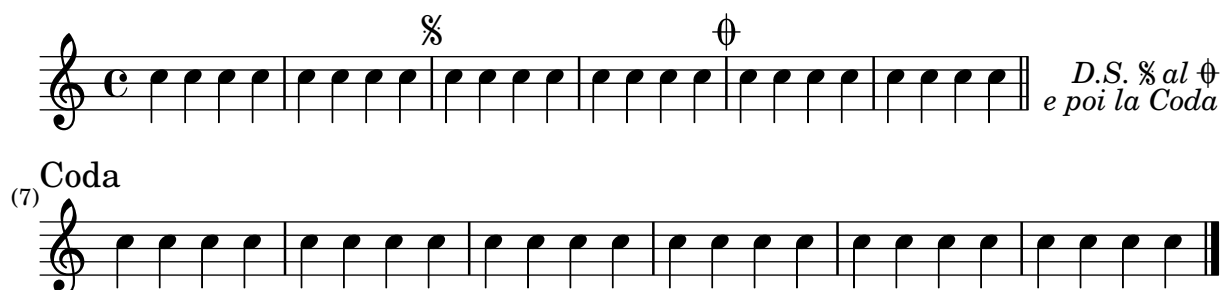
\relative c' {
  c4 c c c | c c c c |
  \repeat segno 2 {
    c4 c c c | c c c c |
    \alternative {
      \volta 1 {
        c4 c c c | c c c c |
        % If you don't use \break at Coda, use \noBreak here
        % and after \bar "" below.
        \noBreak
        \section % double bar line
        \cadenzaOn % pause bar count
        \stopStaff % remove staff lines
        % Increasing the unfold counter will expand the staff-free space
        \repeat unfold 4 {
          s1
          \bar ""
        }
      }
    }
  }
}

```

```

    % Place JumpScript where the staff would normally be.
    \once \override Score.JumpScript.outside-staff-priority = ##f
    \once \override Score.JumpScript.Y-offset = 0
    \startStaff % resume bar count
    \cadenzaOff % show staff lines again
  }
}
}
\sectionLabel "Coda"
% Show Coda on a new line
\break
\repeat unfold 6 { c4 c c c }
\fine
}

```



## Rest styles

Rests may be used in various styles.

```

restsA = {
  r\maxima r\longa r\breve r1 r2 r4 r8 r16 s32
  s64 s128 s256 s512 s1024 s1024
}

restsB = {
  r\maxima r\longa r\breve r1 r2 r4 r8 r16 r32
  r64 r128 r256 r512 r1024 s1024
}

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

  \override Staff.Rest.style = #'mensural
  <>^\markup \typewriter { mensural } \restsA \bar "" \break

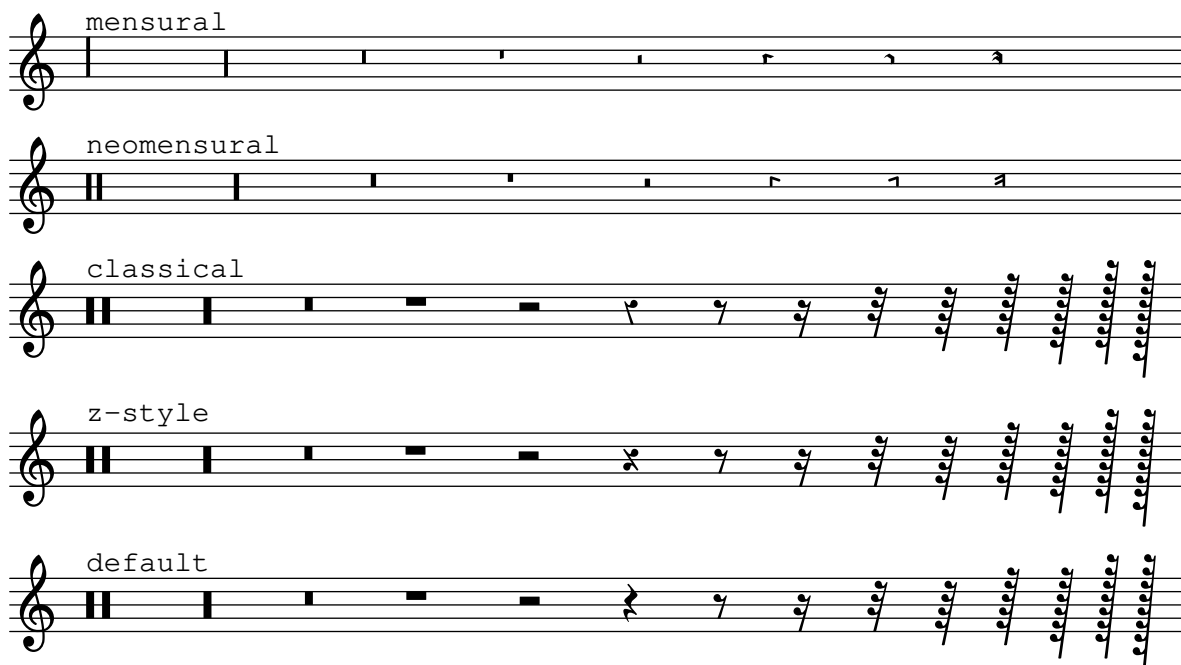
  \override Staff.Rest.style = #'neomensural
  <>^\markup \typewriter { neomensural } \restsA \bar "" \break

  \override Staff.Rest.style = #'classical
  <>^\markup \typewriter { classical } \restsB \bar "" \break

  \override Staff.Rest.style = #'z
  <>^\markup \typewriter { z-style } \restsB \bar "" \break
}

```

```
\override Staff.Rest.style = #'default
<>~\markup \typewriter { default } \restsB \bar "" \break
}
```



## Volta text markup using repeatCommands

Though voltes are best specified using `\repeat volta`, the context property `repeatCommands` must be used in cases where the volta text needs more advanced formatting with `\markup`.

Since `repeatCommands` takes a list, the simplest method of including markup is to use an identifier for the text and embed it in the command list using the Scheme syntax  `#(list (list 'volta textIdentifier))`. Start- and end-repeat commands can be added as separate list elements:

```
voltaAdLib = \markup { \volta-number { 1. 2. 3... } \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))
}
```



## 36 Templates

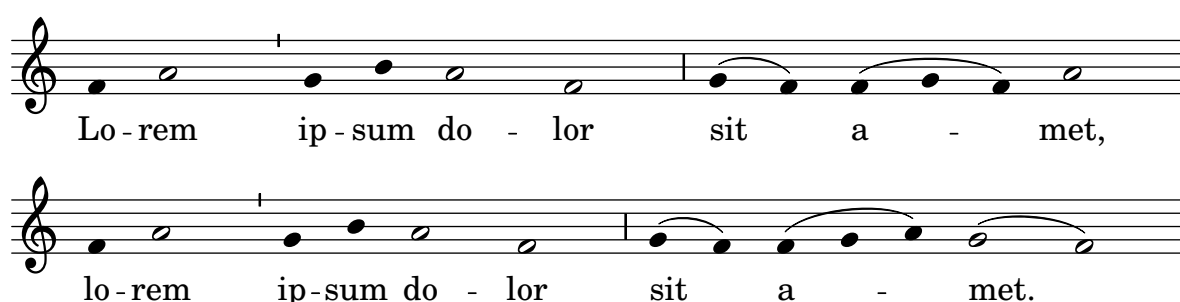
### Ancient notation template – modern transcription of Gregorian music

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.

```
chant = \relative c' {
  \set Score.timing = ##f
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g f) a2 \finalis \break
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g a) g2( f) \finalis
}

verba = \lyricmode {
  Lo -- rem ip -- sum do -- lor sit a -- met,
  lo -- rem ip -- sum do -- lor sit a -- met.
}

\score {
  \new GregorianTranscriptionStaff <<
    \new GregorianTranscriptionVoice = "melody" \chant
    \new GregorianTranscriptionLyrics = "one" \lyricsto melody \verba
  >>
}
```



### Anglican psalm template

This template shows one way of setting out an Anglican psalm chant. It also shows how the verses may be added as stand-alone text under the music. The two verses are coded in different styles to demonstrate more possibilities.

```
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
}

dot = \markup {
    \raise #0.7 \musicglyph "dots.dot"
}

tick = \markup {
    \raise #1 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}

% Use markup to center the chant on the page
\markup \fill-line {
    \score { % centered
        \new ChoirStaff <<
            \new Staff <<
                \global
                \clef "treble"
                \new Voice = "Soprano" <<
                    \voiceOne
                    \SopranoMusic
                >>
                \new Voice = "Alto" <<
                    \voiceTwo
                    \AltoMusic
                >>
            >>

            \new Staff <<
                \clef "bass"
                \global
                \new Voice = "Tenor" <<
                    \voiceOne
                    \TenorMusic
                >>
                \new Voice = "Bass" <<
                    \voiceTwo
                    \BassMusic
            >>
        >>
    }
}

```

```

>>
>>
>>

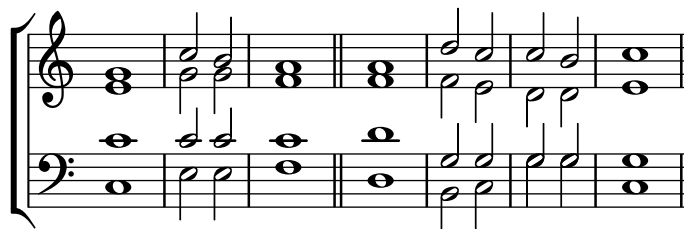
\layout {
  \context {
    \Score
    \override SpacingSpanner.base-shortest-duration =
      \musicLength 2
  }
  \context {
    \Staff
    \remove "Time_signature_engraver"
  }
}
} % End score
} % End markup

\markup \fill-line {
  \left-column {
    \null \null \null
    \line { \fontsize #5 0
      \fontsize #3 come
      let us \bold sing | unto \dot the | Lord : let }
    \line { us heartily \concat { re \bold joyce }
      in the | strength of | our }
    \line { sal | vation. }

    \null

    \line { \hspace #2.5 8. Today if ye will hear his voice * }
    \line { \concat { \bold hard en }
      \tick not your \tick hearts : as in the pro- }
    \line { vocation * and as in the \bold day of tempt- \tick }
    \line { -ation \tick in the \tick wilderness. }
  }
}

```



**O** come let us **sing** | unto • the | Lord : let  
us heartily **rejoice** in the | strength of | our  
sal | vation.

8. Today if ye will hear his voice \*  
**harden** ' not your ' hearts : as in the pro-  
vocation \* and as in the **day** of tempt- '  
-ation ' in the ' wilderness.

## Hymn template

This code shows one way of setting out a hymn tune where 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 \caesura \break
  s2 | s1 | s2 \breathe s2 | s1 | s2 \fine
}

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 {
  \left-column {
    "This is line one of the first verse"
    "This is line two of the same"
  }
  \null
  "And here's line one of the second verse"
  "And the next line of the same"
}

\layout {
  \context {
    \Score
    caesuraType = #'((bar-line . "||"))
    fineBarType = "||"
  }
}

\paper { % Start paper block
  indent = 0 % don't indent first system
  line-width = 130 % shorten line length to suit music

```

```

tagline = ##f      % Don't print tag line, can be removed
} % End paper block

```



This is line one of the first verse  
This is line two of the same

And here's line one of the second verse  
And the next line of the same

## Jazz combo template

This is quite an advanced template, for a jazz ensemble. Note that all instruments use `\key c \major`. This refers to the key in concert pitch; the key will be automatically transposed if the music is within a `\transpose` section.

```

\header {
  title = "Song"
  subtitle = "(tune)"
  composer = "Me"
  meter = "moderato"
  piece = "Swing"
  tagline = \markup \column {
    "LilyPond example file by Amelie Zapf,"
    "Berlin 07/07/2003" }
}

% To make the example display properly in the documentation.
\paper {
  paper-width = 130\mm
  paper-height = 205\mm
}

% #(set-global-staff-size 16)

\include "english.ly"

%%%%%%%%%% Some macros %%%%%%%%%%

```

```

sl = { \override NoteHead.style = #'slash
      \hide Stem }
nsl = { \revert NoteHead.style
      \undo \hide Stem }
crOn = \override NoteHead.style = #'cross
crOff = \revert NoteHead.style

% Insert chord name style stuff here.

jazzChords = { }

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Keys 'n' thangs %%%%%%%%%%%%%%%

global = { \time 4/4 }

Key = { \key c \major }

% ##### Horns #####

% ----- Trumpet -----
trpt = \transpose c d \relative c' {
  \Key
  c1 | c | c |
}
trpHarmony = \transpose c' d {
  \jazzChords
}
trumpet = {
  \global
  \clef treble
  \trpt
}

% ----- Alto Saxophone -----
alto = \transpose c a \relative c' {
  \Key
  c1 | c | c |
}
altoHarmony = \transpose c' a {
  \jazzChords
}
altoSax = {
  \global
  \clef treble
  \alto
}

% ----- Baritone Saxophone -----
bari = \transpose c a' \relative c {
  \Key
  c1 | c1 |
}

```

```

    \sl d4^"Solo" d d d \ns1 |
}
bariHarmony = \transpose c' a \chordmode {
    \jazzChords
    s1 | s |
    d2:maj e:m7 |
}
bariSax = {
    \global
    \clef treble
    \bari
}

% ----- Trombone -----
tbone = \relative c {
    \Key
    c1 | c | c |
}
tboneHarmony = \chordmode {
    \jazzChords
}
trombone = {
    \global
    \clef bass
    \tbone
}

% ##### Rhythm Section #####

% ----- Guitar -----
gtr = \relative c'' {
    \Key
    c1 |
    \sl b4 b b b \ns1 |
    c1 |
}
gtrHarmony = \chordmode {
    \jazzChords
    s1 | c2:min7+ d2:maj9 | s1 |
}
guitar = {
    \global
    \clef treble
    \gtr
}

%% ----- Piano -----
rhUpper = \relative c'' {
    \voiceOne
    \Key
    c1 | c | c |
}

```

```

rhLower = \relative c' {
  \voiceTwo
  \Key
  e1 | e | e |
}

lhUpper = \relative c' {
  \voiceOne
  \Key
  g1 | g | g |
}

lhLower = \relative c {
  \voiceTwo
  \Key
  c1 | c | c |
}

PianoRH = {
  \clef treble
  \global
  <<
    \new Voice = "one" \rhUpper
    \new Voice = "two" \rhLower
  >>
}

PianoLH = {
  \clef bass
  \global
  <<
    \new Voice = "one" \lhUpper
    \new Voice = "two" \lhLower
  >>
}

piano = <<
  \new Staff = "upper" \PianoRH
  \new Staff = "lower" \PianoLH
>>

% ----- Bass Guitar -----
Bass = \relative c {
  \Key
  c1 | c | c |
}

bass = {
  \global
  \clef bass
  \Bass
}

% ----- Drums -----

```



```

up = \drummode {
  \voiceOne
  hh4 <hh sn> hh <hh sn> |
  hh4 <hh sn> hh <hh sn> |
  hh4 <hh sn> hh <hh sn> |
}
down = \drummode {
  \voiceTwo
  bd4 s bd s |
  bd4 s bd s |
  bd4 s bd s |
}

drumContents = {
  \global
  <<
    \new DrumVoice \up
    \new DrumVoice \down
  >>
}

%%%%%%%%%% It All Goes Together Here %%%%%%%%%%%%%%

\book { % For the LilyPond documentation.
  \score {
    <<
      \new StaffGroup = "horns" <<
        \new Staff = "trumpet" \with { instrumentName = "Trumpet" }
        \trumpet
        \new Staff = "altosax" \with { instrumentName = "Alto Sax" }
        \altoSax
        \new ChordNames = "barichords" \with { instrumentName = "Bari Sax" }
        \bariHarmony
        \new Staff = "barisax" \with { instrumentName = "Bari Sax" }
        \bariSax
        \new Staff = "trombone" \with { instrumentName = "Trombone" }
        \trombone
      >>

      \new StaffGroup = "rhythm" <<
        \new ChordNames = "chords" \with { instrumentName = "Guitar" }
        \gtrHarmony
        \new Staff = "guitar" \with { instrumentName = "Guitar" }
        \guitar
        \new PianoStaff = "piano" \with {
          instrumentName = "Piano"
          midiInstrument = "acoustic grand"
        } \piano
        \new Staff = "bass" \with { instrumentName = "Bass" }
        \bass
        \new DrumStaff \with { instrumentName = "Drums" }
        \drumContents
    >>
  }
}

```

```
>>
>>

\layout {
  \context {
    \Staff
    \RemoveEmptyStaves
  }
  \context {
    \Score
    \override BarNumber.padding = 3
    \override RehearsalMark.padding = 2
    skipBars = ##t
  }
}
\midi { }
}
```

## Song (tune)

Me

moderato  
Swing

Trumpet

Alto Sax

Bari Sax

Trombone

Guitar

Piano

Bass

Drums

B<sup>Δ</sup> C<sup>#</sup>m<sup>7</sup>  
Solo

Cm<sup>Δ</sup> D<sup>Δ</sup>9

LilyPond example file by Amelie Zapf,  
Berlin 07/07/2003

### Orchestra, choir and piano template

This template demonstrates the use of nested `StaffGroup` and `GrandStaff` contexts to subgroup instruments of the same type together, and a way to use `\transpose` so that variables hold music for transposing instruments at concert pitch.

```
#(set-global-staff-size 17)
```

```
\paper {
  indent = 3.0\cm % add space for instrumentName
  short-indent = 1.5\cm % add less space for shortInstrumentName
}
```

```

fluteMusic = \relative c' { \key g \major g'1 b }

% Pitches as written on a manuscript for Clarinet in A
% are transposed to concert pitch.
clarinetMusic = \transpose c' a
  \relative c'' { \key bes \major bes1 d }

trumpetMusic = \relative c { \key g \major g''1 b }

% Key signature is often omitted for horns
hornMusic = \transpose c' f
  \relative c { d'1 fis }

percussionMusic = \relative c { \key g \major g1 b }

sopranoMusic = \relative c'' { \key g \major g'1 b }
sopranoLyrics = \lyricmode { Lyr -- ics }

altoIMusic = \relative c' { \key g \major g'1 b }
altoILyrics = \sopranoLyrics
altoIIMusic = \relative c' { \key g \major g'1 b }
altoIILyrics = \lyricmode { Ah -- ah }

tenorMusic = \relative c' { \clef "treble_8" \key g \major g1 b }
tenorLyrics = \sopranoLyrics

pianoRHMus = \relative c { \key g \major g''1 b }
pianoLHMus = \relative c { \clef bass \key g \major g1 b }

violinIMusic = \relative c' { \key g \major g'1 b }
violinIIMusic = \relative c' { \key g \major g'1 b }

violaMusic = \relative c { \clef alto \key g \major g'1 b }

celloMusic = \relative c { \clef bass \key g \major g1 b }

bassMusic = \relative c { \clef "bass_8" \key g \major g,1 b }

\book {
  \score {
    <<
    \new StaffGroup = "StaffGroup_woodwinds" <<
      \new Staff = "Staff_flute" \with { instrumentName = "Flute" }
        \fluteMusic

      \new Staff = "Staff_clarinet" \with {
        instrumentName = \markup { \concat { "Clarinet in B" \flat } }
      }

      % Declare that written Middle C in the music
      % to follow sounds a concert B flat, for
      % output using sounded pitches such as MIDI.
      %\transposition bes

```

```

    % Print music for a B-flat clarinet
    \transpose bes c' \clarinetMusic
>>

\new StaffGroup = "StaffGroup_brass" <<
  \new Staff = "Staff_hornI" \with {
    instrumentName = "Horn in F"
  }
  % \transposition f
  \transpose f c' \hornMusic

  \new Staff = "Staff_trumpet" \with {
    instrumentName = "Trumpet in C"
  }
  \trumpetMusic
>>

\new RhythmicStaff = "RhythmicStaff_percussion" \with {
  instrumentName = "Percussion"
}
  \percussionMusic

\new PianoStaff \with {
  instrumentName = "Piano"
} <<
  \new Staff { \pianoRHMusical }
  \new Staff { \pianoLHMusical }
>>

\new ChoirStaff = "ChoirStaff_choir" <<
  \new Staff = "Staff_soprano" \with {
    instrumentName = "Soprano"
  }
  \new Voice = "soprano" \sopranoMusical
  \new Lyrics \lyricsto "soprano" { \sopranoLyrics }

  \new GrandStaff = "GrandStaff_alto" \with {
    \accepts Lyrics
  } <<
  \new Staff = "Staff_altoI" \with {
    instrumentName = "Alto I"
  }
  \new Voice = "altoI"
  \altoIMusical
  \new Lyrics \lyricsto "altoI" { \altoILyrics }
  \new Staff = "Staff_altoII" \with {
    instrumentName = "Alto II"
  }
  \new Voice = "altoII"
  \altoIIMusical
  \new Lyrics \lyricsto "altoII" { \altoIILyrics }

```

```

>>

\new Staff = "Staff_tenor" \with {
  instrumentName = "Tenor"
}
  \new Voice = "tenor" \tenorMusic
  \new Lyrics \lyricsto "tenor" { \tenorLyrics }
>>

\new StaffGroup = "StaffGroup_strings" <<
  \new GrandStaff = "GrandStaff_violins" <<
    \new Staff = "Staff_violinI" \with {
      instrumentName = "Violin I"
    }
      \violinIMusic
    \new Staff = "Staff_violinII" \with {
      instrumentName = "Violin II"
    }
      \violinIIMusic
  >>
>>

\new Staff = "Staff_viola" \with {
  instrumentName = "Viola"
}
  \violaMusic

\new Staff = "Staff_cello" \with {
  instrumentName = "Cello"
}
  \celloMusic

\new Staff = "Staff_bass" \with {
  instrumentName = "Double Bass"
}
  \bassMusic
>>
>>
}
}

```

Flute

Clarinet in B $\flat$

Horn in F

Trumpet in C

Percussion

Piano

Soprano

Alto I

Alto II

Tenor

Violin I

Violin II

Viola

Cello

Double Bass

Lyr - ics

Lyr - ics

Ah - ah

Lyr - ics

## Piano template (simple)

Here is a simple piano staff with some notes.

```
upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  a2 c
}

\score {
  \new PianoStaff \with { instrumentName = "Piano" }
  <<
    \new Staff = "upper" \upper
    \new Staff = "lower" \lower
  >>
  \layout { }
  \midi { }
}
```



## Piano template with centered lyrics

Instead of having a full staff for the melody and lyrics, lyrics can be centered between the staves of a piano staff.

```
upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4
```



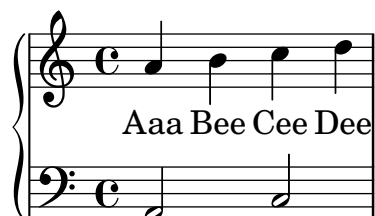
```

    a2 c
}

text = \lyricmode {
    Aaa Bee Cee Dee
}

\score {
  \new PianoStaff <<
    \new Staff = upper { \new Voice = "singer" \upper }
    \new Lyrics \lyricsto "singer" \text
    \new Staff = lower { \lower }
  >>
  \layout { }
  \midi { }
}

```



## Piano template with melody and lyrics

Here is a typical song format: one staff with the melody and lyrics, with piano accompaniment underneath.

```

melody = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

upper = \relative c'' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

lower = \relative c {
  \clef bass
  \key c \major
  \time 4/4
}

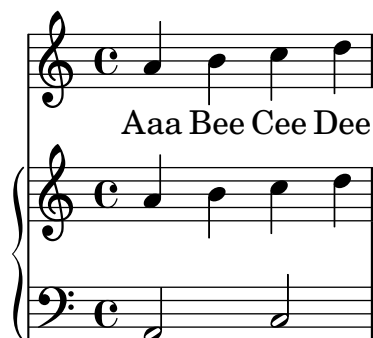
```

```

a2 c
}

\score {
  <<
    \new Voice = "mel" { \autoBeamOff \melody }
    \new Lyrics \lyricsto mel \text
    \new PianoStaff <<
      \new Staff = "upper" \upper
      \new Staff = "lower" \lower
    >>
  >>
  \layout {
    \context { \Staff \RemoveEmptyStaves }
  }
  \midi { }
}

```



## SATB choir template – four staves

This is a template for a SATB choir on four staves.

```

global = {
  \key c \major
  \time 4/4
  \dynamicUp
}
sopranonotes = \relative c'' {
  c2 \p \< d c d \f
}
sopranowords = \lyricmode { do do do do }
altonotes = \relative c'' {
  c2\p d c d
}
altowords = \lyricmode { re re re re }
tenornotes = {
  \clef "G_8"
  c2\mp d c d
}
tenorwords = \lyricmode { mi mi mi mi }
bassnotes = {
  \clef bass

```

```

    c2\mf d c d
}
basswords = \lyricmode { mi mi mi mi }

\score {
  \new ChoirStaff <<
    \new Staff <<
      \new Voice = "soprano" <<
        \global
        \sopranonotes
      >>
      \new Lyrics \lyricsto "soprano" \sopranowords
    >>
    \new Staff <<
      \new Voice = "alto" <<
        \global
        \altonotes
      >>
      \new Lyrics \lyricsto "alto" \altowords
    >>
    \new Staff <<
      \new Voice = "tenor" <<
        \global
        \tenornotes
      >>
      \new Lyrics \lyricsto "tenor" \tenorwords
    >>
    \new Staff <<
      \new Voice = "bass" <<
        \global
        \bassnotes
      >>
      \new Lyrics \lyricsto "bass" \basswords
    >>
  >>
}

```



## Single-staff template with notes, lyrics, and chords

This template allows the preparation of a song with melody, words, and chords.

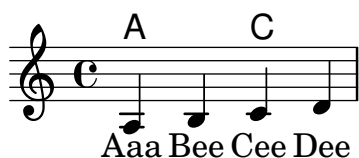
```
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

harmonies = \chordmode {
  a2 c
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \harmonies
    }
    \new Voice = "one" { \autoBeamOff \melody }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
  \midi { }
}
```



## Single-staff template with notes, lyrics, chords, and frets

Here is a simple lead sheet template with melody, lyrics, chords, and fret diagrams.

```

verseI = \lyricmode {
  \set stanza = #"1."
  This is the first verse
}

verseII = \lyricmode {
  \set stanza = #"2."
  This is the second verse.
}

theChords = \chordmode {
  % insert chords for chordnames and fretboards here
  c2 g4 c
}

staffMelody = \relative c' {
  \key c \major
  \clef treble
  % Type notes for melody here
  c4 d8 e f4 g
  \bar "|"
}

\score {
  <<
    \context ChordNames { \theChords }
    \context FretBoards { \theChords }
    \new Staff {
      \context Voice = "voiceMelody" { \staffMelody }
    }
    \new Lyrics = "lyricsI" {
      \lyricsto "voiceMelody" \verseI
    }
    \new Lyrics = "lyricsII" {
      \lyricsto "voiceMelody" \verseII
    }
  >>
  \layout { }
  \midi { }
}

```

1. This is the first verse  
2. This is the second verse.

## Single-staff template with notes and chords

Want to prepare a lead sheet with a melody and chords? Look no further!

```
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  f4 e8[ c] d4 g |
  a2 ~ a
}

harmonies = \chordmode {
  c4:m f:min7 g:maj c:aug |
  d2:dim b4:5 e:sus
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \harmonies
    }
    \new Staff \melody
  >>
  \layout{ }
  \midi { }
}
```

## Single-staff template with notes and lyrics

This small template demonstrates a simple melody with lyrics. Cut and paste, add notes, then words for the lyrics. This example turns off automatic beaming, which is common for vocal parts. To use automatic beaming, change or comment out the relevant line.

```
melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4
```

```

a4 b c d
}

text = \lyricmode {
  Aaa Bee Cee Dee
}

\score{
  <<
    \new Voice = "one" {
      \autoBeamOff
      \melody
    }
    \new Lyrics \lyricsto "one" \text
  >>
  \layout { }
  \midi { }
}

```



## Single-staff template with only notes

This very simple template gives you a staff with notes, suitable for a solo instrument or a melodic fragment. Cut and paste this into a file, add notes, and you're finished!

```

melody = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

```

```

\score {
  \new Staff \melody
  \layout { }
  \midi { }
}

```



## String quartet template (simple)

This template demonstrates a simple string quartet. It also uses a `\global` section for time and key signatures.

See also snippet “String quartet template with separate parts”.

```

global= {

```

```

\time 4/4
\key c \major
}

violinOne = \new Voice \relative c' {
  c2 d
  e1
  \bar "|"
}

violinTwo = \new Voice \relative c' {
  g2 f
  e1
  \bar "|"
}

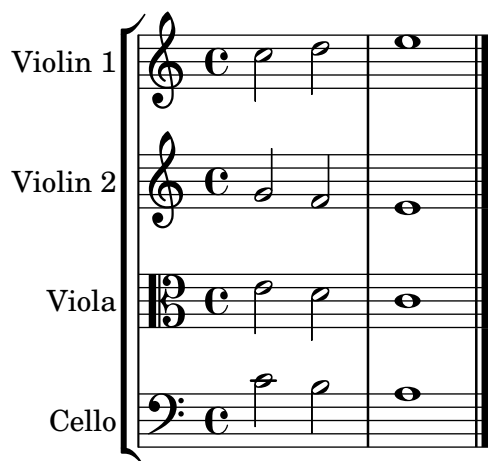
viola = \new Voice \relative c' {
  \clef alto
  e2 d
  c1
  \bar "|"
}

cello = \new Voice \relative c' {
  \clef bass
  c2 b
  a1
  \bar "|"
}

\score {
  \new StaffGroup <<
    \new Staff \with { instrumentName = "Violin 1" }
    << \global \violinOne >>
    \new Staff \with { instrumentName = "Violin 2" }
    << \global \violinTwo >>
    \new Staff \with { instrumentName = "Viola" }
    << \global \viola >>
    \new Staff \with { instrumentName = "Cello" }
    << \global \cello >>
  >>
  \layout { }
  \midi { }
}

```





## String quartet template with separate parts

The “String quartet template (simple)” snippet produces a nice string quartet, but what if you need to print parts? This new template demonstrates how to use the `\tag` feature to easily split a piece into individual parts.

For technical reasons, multiple output files cannot be shown here for a single snippet, which means that the template below unifies the code for separate files. The file names are contained in comments at the beginning of each file.

`piece.ly` contains all the music definitions. The other files – `score.ly`, `vn1.ly`, `vn2.ly`, `vla.ly`, and `vlc.ly` – produce the full score and the four parts.

Do not forget to remove specified comments when using separate files!

```
% piece.ly
% (This is the global definitions file.)
```

```
global= {
  \time 4/4
  \key c \major
}
```

```
Violinone = \new Voice \relative c' {
  c2 d e1
  \bar "|."
}
```

```
Violintwo = \new Voice \relative c' {
  g2 g e1
  \bar "|."
}
```

```
Viola = \new Voice \relative c' {
  \clef alto
  e2 d c1
  \bar "|."
}
```

```
Cello = \new Voice \relative c' {
  \clef bass
```

```

    c2 b a1
    \bar "|"
}

music = <<
  \tag #'score \tag #'vn1
  \new Staff \with { instrumentName = "Violin 1" }
    << \global \Violinone >>

  \tag #'score \tag #'vn2
  \new Staff \with { instrumentName = "Violin 2" }
    << \global \Violintwo >>

  \tag #'score \tag #'vla
  \new Staff \with { instrumentName = "Viola" }
    << \global \Viola >>

  \tag #'score \tag #'vlc
  \new Staff \with { instrumentName = "Cello" }
    << \global \Cello >>
>>

% These are the other files you need to save on your computer

% score.ly
% (This is the main file.)

% Uncomment the line below when using a separate file.
% \include "piece.ly"

#(set-global-staff-size 14)

\score {
  \new StaffGroup \keepWithTag #'score \music
  \layout { }
  \midi { }
}

%{ Uncomment this block when using separate files.

% vn1.ly
% (This is the Violin 1 part file.)

\include "piece.ly"
\score {
  \keepWithTag #'vn1 \music
  \layout { }
}

% vn2.ly

```

```
% (This is the Violin 2 part file.)
```

```
\include "piece.ly"
\score {
  \keepWithTag #'vn2 \music
  \layout { }
}
```

```
% vla.ly
% (This is the Viola part file.)
```

```
\include "piece.ly"
\score {
  \keepWithTag #'vla \music
  \layout { }
}
```

```
% vlc.ly
% (This is the Cello part file.)
```

```
\include "piece.ly"
\score {
  \keepWithTag #'vlc \music
  \layout { }
}
```

```
%}
```

## Vocal ensemble template

Here is a standard four-part SATB vocal score. With larger ensembles, it is often useful to include a section which is included in all parts. For example, the time signature and key signature are almost always the same for all parts. Like in the “Hymn template”, the four voices are regrouped on only two staves.

```
\paper {
  top-system-spacing.basic-distance = 10
  score-system-spacing.basic-distance = 20
  system-system-spacing.basic-distance = 20
  last-bottom-spacing.basic-distance = 10
}
```

```
global = {
```

```

\key c \major
\time 4/4
}

sopMusic = \relative {
  c''4 c c8[( b)] c4
}
sopWords = \lyricmode {
  hi hi hi hi
}

altoMusic = \relative {
  e'4 f d e
}
altoWords = \lyricmode {
  ha ha ha ha
}

tenorMusic = \relative {
  g4 a f g
}
tenorWords = \lyricmode {
  hu hu hu hu
}

bassMusic = \relative {
  c4 c g c
}
bassWords = \lyricmode {
  ho ho ho ho
}

\score {
  \new ChoirStaff <<
    \new Lyrics = "sopranos" \with {
      % this is needed for lyrics above a staff
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
    \new Staff = "women" <<
      \new Voice = "sopranos" {
        \voiceOne
        << \global \sopMusic >>
      }
      \new Voice = "altos" {
        \voiceTwo
        << \global \altoMusic >>
      }
    >>
    \new Lyrics = "altos"
    \new Lyrics = "tenors" \with {
      % this is needed for lyrics above a staff
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
  >>
}

```

```

}
\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" \sopWords
\context Lyrics = "altos" \lyricsto "altos" \altoWords
\context Lyrics = "tenors" \lyricsto "tenors" \tenorWords
\context Lyrics = "basses" \lyricsto "basses" \bassWords
>>
}

```



## Vocal ensemble template with automatic piano reduction

This template adds an automatic piano reduction to the standard SATB vocal score demonstrated in snippet “Vocal ensemble template”. It demonstrates one of the strengths of LilyPond – you can use a music definition more than once. If any changes are made to the vocal notes (say, `tenorMusic`), then the changes also apply to the piano reduction.

```

\paper {
  top-system-spacing.basic-distance = 10
  score-system-spacing.basic-distance = 20
  system-system-spacing.basic-distance = 20
  last-bottom-spacing.basic-distance = 10
}

global = {
  \key c \major
  \time 4/4
}

sopMusic = \relative {
  c' '4 c c8[( b)] c4
}

sopWords = \lyricmode {

```

```

    hi hi hi hi
}

altoMusic = \relative {
    e'4 f d e
}
altoWords = \lyricmode {
    ha ha ha ha
}

tenorMusic = \relative {
    g4 a f g
}
tenorWords = \lyricmode {
    hu hu hu hu
}

bassMusic = \relative {
    c4 c g c
}
bassWords = \lyricmode {
    ho ho ho ho
}

\score {
  <<
    \new ChoirStaff <<
      \new Lyrics = "sopranos" \with {
        % This is needed for lyrics above a staff
        \override VerticalAxisGroup.staff-affinity = #DOWN
      }
      \new Staff = "women" <<
        \new Voice = "sopranos" { \voiceOne << \global \sopMusic >> }
        \new Voice = "altos" { \voiceTwo << \global \altoMusic >> }
      >>
      \new Lyrics = "altos"

      \new Lyrics = "tenors" \with {
        % This is needed for lyrics above a staff
        \override VerticalAxisGroup.staff-affinity = #DOWN
      }
      \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" \sopWords
      \context Lyrics = "altos" \lyricsto "altos" \altoWords
      \context Lyrics = "tenors" \lyricsto "tenors" \tenorWords
      \context Lyrics = "basses" \lyricsto "basses" \bassWords
    >>
  >>
}

```

```

>>

\new PianoStaff <<
  \new Staff <<
    \set Staff.printPartCombineTexts = ##f
    \partCombine
    << \global \sopMusic >>
    << \global \altoMusic >>
  >>
  \new Staff <<
    \clef bass
    \set Staff.printPartCombineTexts = ##f
    \partCombine
    << \global \tenorMusic >>
    << \global \bassMusic >>
  >>
>>
>>
>>
}

```

hi hi hi hi

ha ha ha ha

hu hu hu hu

ho ho ho ho

## Vocal ensemble template with lyrics aligned below and above the staves

This template is basically the same as the simple “Vocal ensemble template”, with the exception that here all the lyrics lines are placed using `alignAboveContext` and `alignBelowContext`.

```

global = {
  \key c \major
  \time 4/4
}

sopMusic = \relative c'' {
  c4 c c8[( b)] c4
}

```

```

sopWords = \lyricmode {
  hi hi hi hi
}

altoMusic = \relative c' {
  e4 f d e
}
altoWords = \lyricmode {
  ha ha ha ha
}

tenorMusic = \relative c' {
  g4 a f g
}
tenorWords = \lyricmode {
  hu hu hu hu
}

bassMusic = \relative c {
  c4 c g c
}
bassWords = \lyricmode {
  ho ho ho ho
}

\score {
  \new ChoirStaff <<
    \new Staff = "women" <<
      \new Voice = "sopranos" { \voiceOne << \global \sopMusic >> }
      \new Voice = "altos" { \voiceTwo << \global \altoMusic >> }
    >>
    \new Lyrics \with { alignAboveContext = "women" }
      \lyricsto "sopranos" \sopWords
    \new Lyrics \with { alignBelowContext = "women" }
      \lyricsto "altos" \altoWords
    % we could remove the line about this with the line below, since
    % we want the alto lyrics to be below the alto Voice anyway.
    % \new Lyrics \lyricsto "altos" \altoWords

    \new Staff = "men" <<
      \clef bass
      \new Voice = "tenors" { \voiceOne << \global \tenorMusic >> }
      \new Voice = "basses" { \voiceTwo << \global \bassMusic >> }
    >>
    \new Lyrics \with { alignAboveContext = "men" }
      \lyricsto "tenors" \tenorWords
    \new Lyrics \with { alignBelowContext = "men" }
      \lyricsto "basses" \bassWords
    % again, we could replace the line above this with the line below.
    % \new Lyrics \lyricsto "basses" \bassWords
  >>
}

```





## Vocal ensemble template with verse and refrain

This template creates a score that starts with a solo verse and continues into a refrain for two voices. It also demonstrates the use of spacer rests within the `\global` variable to define meter changes (and other elements common to all parts) throughout the entire score.

```

global = {
  \key g \major

  % verse
  \time 3/4
  s2.*2
  \break

  % refrain
  \time 2/4
  s2*2
  \bar "|"
}

SoloNotes = \relative g' {
  \clef "treble"

  % verse
  g4 g g |
  b4 b b |

  % refrain
  R2*2 |
}

SoloLyrics = \lyricmode {
  One two three |
  four five six |
}

SopranoNotes = \relative c'' {
  \clef "treble"

  % verse
  R2.*2 |

```

```

    % refrain
    c4 c |
    g4 g |
}

SopranoLyrics = \lyricmode {
    la la |
    la la |
}

BassNotes = \relative c {
    \clef "bass"

    % verse
    R2.*2 |

    % refrain
    c4 e |
    d4 d |
}

BassLyrics = \lyricmode {
    dum dum |
    dum dum |
}

\score {
  <<
    \new Voice = "SoloVoice" << \global \SoloNotes >>
    \new Lyrics \lyricsto "SoloVoice" \SoloLyrics

    \new ChoirStaff <<
      \new Voice = "SopranoVoice" << \global \SopranoNotes >>
      \new Lyrics \lyricsto "SopranoVoice" \SopranoLyrics

      \new Voice = "BassVoice" << \global \BassNotes >>
      \new Lyrics \lyricsto "BassVoice" \BassLyrics
    >>
  >>

  \layout {
    ragged-right = ##t
    \context { \Staff
      % these lines prevent empty staves from being printed
      \RemoveEmptyStaves
      \override VerticalAxisGroup.remove-first = ##t
    }
  }
}

```

One two three four five six

la la la la

dum dum dum dum

The image shows two musical staves. The top staff is a single treble clef in 3/4 time, with a key signature of one sharp (F#). It contains six quarter notes: G4, A4, B4, C5, B4, and A4. Below the staff are the lyrics 'One two three four five six'. The bottom staff is a grand staff (treble and bass clefs) in 3/4 time, with a key signature of one sharp (F#). It contains four measures, each with a quarter note in both hands: G4 in the treble and F#3 in the bass. Below the staff are the lyrics 'la la la la' for the treble part and 'dum dum dum dum' for the bass part. A brace on the left side of the grand staff indicates it is a single musical unit.

## 37 Titles

See also Section “Titles and headers” in *Notation Reference*.

### Adding the current date to a score

With a little Scheme code, the current date can easily be added to a score.

```
\paper { tagline = ##f }

% first, define a variable to hold the formatted date:
date = #(strftime "%d-%m-%Y" (localtime (current-time)))

% use it in the title block:
\header {
  title = "Including the date!"
  subtitle = \date
}

\score {
  \relative c' {
    c4 c c c
  }
}

% and use it in a \markup block:
\markup {
  \date
}
```

### Including the date!

17-01-2026



17-01-2026

### Aligning and centering instrument names

The horizontal alignment of instrument names is tweaked by changing the `self-alignment-X` property of the `InstrumentName` grob (usually in the `Staff` context). The `\layout` variables `indent` and `short-indent` define the space in which the instrument names are aligned before the first and the following systems, respectively.

```
\paper {
  left-margin = 3\cm
}

\new StaffGroup <<
  \new Staff \with {
    \override InstrumentName.self-alignment-X = #LEFT
    instrumentName = \markup \left-column { "Left aligned"
                                              "instrument name" }
    shortInstrumentName = "Left"
```

```

} {
  c''1 \break c''1
}

\new Staff \with {
  \override InstrumentName.self-alignment-X = #CENTER
  instrumentName = \markup \center-column { Centered
                                         "instrument name" }

  shortInstrumentName = "Centered"
} {
  g'1 g'1
}

\new Staff \with {
  \override InstrumentName.self-alignment-X = #RIGHT
  instrumentName = \markup \right-column { "Right aligned"
                                         "instrument name" }

  shortInstrumentName = "Right"
} {
  e'1 e'1
}
>>


\layout {
  indent = 4\cm
  short-indent = 2\cm
  line-width = 6.5\cm
}

```

Left aligned  
instrument name

Centered  
instrument name

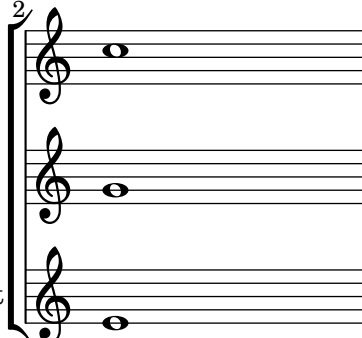
Right aligned  
instrument name



Left

Centered

Right



## Demonstrating all \header fields


A demonstration of all header fields that LilyPond defines by default. Thanks to setting `print-all-headers` to `#t`, much more fields as usual are displayed, indicating the hierarchy of `\header` blocks.

```
\paper {
  #(set-paper-size "a6" 'landscape)
  print-all-headers = ##t
}

\book {
  \header {
    title = "title"
    subtitle = "subtitle"
    composer = "composer"
    arranger = "arranger"
    instrument = "instrument"
    meter = "meter"
    opus = "opus"
    piece = "piece"
    poet = "poet"
    copyright = "copyright"
    tagline = "tagline"
  }

  \bookpart {
    \score {
      \relative c'' { c1 | c | c | c }

      \header {
        title = "localtitle"
        subtitle = "localsubtitle"
        composer = "localcomposer"
        arranger = "localarranger"
        instrument = "localinstrument"
        meter = "localmeter"
        opus = "localopus"
        piece = "localpiece"
        poet = "localpoet"
        copyright = "localcopyright"
        tagline = "localtagline"
      }
    }
  }
}
```

	<b>title</b>	
	<b>subtitle</b>	
poet	<b>instrument</b>	composer
meter		arranger
	<b>localtitle</b>	
	<b>localsubtitle</b>	
localpoet	<b>localinstrument</b>	localcomposer
localmeter		localarranger
localpiece		localopus
		
	copyright	
	tagline	

## Outputting the version number

It is possible to print the version number of LilyPond in markup.

```
\markup { Processed with LilyPond version #(lilypond-version) }
```

Processed with LilyPond version 2.25.32

## 38 Tweaks and overrides

See also Section “Changing defaults” in *Notation Reference* and Section “Tweaking output” in *Learning Manual*.

### Adding an ottava marking to a single voice

If you have more than one voice on the staff, setting octavation in one voice transposes the position of notes in all voices for the duration of the ottava bracket. If the octavation is only intended to apply to one voice, the `Ottava_spanner_engraver` should be moved to `Voice` context.

```
\layout {
  \context {
    \Staff
    \remove Ottava_spanner_engraver
  }
  \context {
    \Voice
    \consists Ottava_spanner_engraver
  }
}

{
  \clef bass
  << { <g d'>1~ q2 <c' e'> }
  \\
  {
    r2.
    \ottava -1
    <b,,, b,,,>4 ~ |
    q2
    \ottava 0
    <c e>2
  }
  >>
}
```



### Adding links to objects

To add a link to a grob stencil you can use `add-link` as defined here. It works both with `\override` and `\tweak`.

Drawback: point-and-click is disturbed for the linked grobs.

Limitation: Works for PDF only.

The linked objects are colored with a separate command.

```
#(define (add-link url-strg)
```



```

(lambda (grob)
  (let* ((stil (ly:grob-property grob 'stencil)))
    (if (ly:stencil? stil)
      (let* ((x-ext (ly:stencil-extent stil X))
              (y-ext (ly:stencil-extent stil Y))
              (url-expr `(url-link ,url-strg ,x-ext ,y-ext))
              (new-stil
                (ly:stencil-add
                 (ly:make-stencil url-expr x-ext y-ext)
                 stil)))
        (ly:grob-set-property! grob 'stencil new-stil))))))

%%% test

%% For easier maintenance of this snippet the URL is formatted to use the
%% actually used LilyPond version.
%% Of course a literal URL would work as well.

#(define major.minor-version
  (string-join (take (string-split (lilypond-version) #\.) 2) "."))

urlI =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/writing-pitches"
  major.minor-version)

urlII =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/rhythms"
  major.minor-version)

urlIII =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/note-heads"
  major.minor-version)

urlIV =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/beams"
  major.minor-version)

urlV =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/note-head-styles"
  major.minor-version)

urlVI =
#(format #f
  "http://lilypond.org/doc/v~a/Documentation/notation/writing-pitches"
  major.minor-version)

\relative c' {

```



```

\clef moderntab
\revert TextScript.stencil
\override TextScript.font-series = #'bold
\override TextScript.font-size = #-2
\override TextScript.color = #red
}
\context {
  \Score
  proportionalNotationDuration = #1/8
}
}

```

	1	&	2	&	3	&	4	&	1	&	2	&	3	&	4	&
<b>T</b>					1	-	1						1	-	1	
<b>A</b>					0	-	0						0	-	0	
<b>B</b>	3				3			2	3				3			2
							3								3	

## 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 |
}
```



## Adjusting grace note spacing

The space given to grace notes can be adjusted using the `spacing-increment` property of `Score.GraceSpacing`.

```
graceNotes = {
  \grace { c4 c8 c16 c32 }
  c8
}

\relative c'' {
  c8
  \graceNotes
  \override Score.GraceSpacing.spacing-increment = #2.0
  \graceNotes
  \revert Score.GraceSpacing.spacing-increment
  \graceNotes
}
```



## Adjusting slur positions vertically

Using `\override Slur.positions` it is possible to set the vertical position of the start and end points of a slur to absolute values (or rather, forcing LilyPond's slur algorithm to consider these values as desired). In many cases, this means a lot of trial and error until good values are found. You probably have tried the `\offset` command next just to find out that it doesn't work for slurs, emitting a warning instead.

The code in this snippet allows you to tweak the vertical start and end positions by specifying *relative* changes, similar to `\offset`.

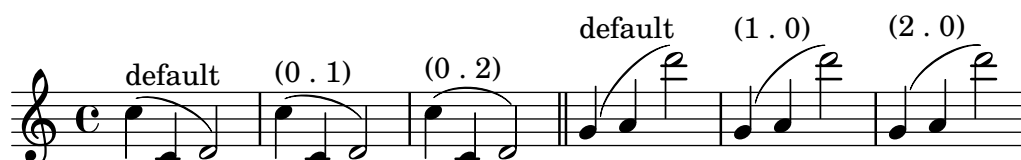
Syntax: `\offsetPositions #'(dy1 . dy2)`

```
offsetPositions =
#(define-music-function (offsets) (number-pair?)
  #{
    \once \override Slur.control-points =
      #(lambda (grob)
        (match-let (((_ . y1) _ _ (_ . y2))
                    (ly:slur::calc-control-points grob))
          ((off1 . off2) offsets))
        (set! (ly:grob-property grob 'positions)
              (cons (+ y1 off1) (+ y2 off2)))
        (ly:slur::calc-control-points grob)))
  #})
```

```

\relative c' {
  c4(^"default" c, d2)
  \offsetPositions #'(0 . 1)
  c'4(^"(0 . 1)" c, d2)
  \offsetPositions #'(0 . 2)
  c'4(^"(0 . 2)" c, d2)
  \bar "||"
  g4(^"default" a d'2)
  \offsetPositions #'(1 . 0)
  g,,4(^"(1 . 0)" a d'2)
  \offsetPositions #'(2 . 0)
  g,,4(^"(2 . 0)" a d'2)
}

```



## Adjusting vertical spacing of lyrics

This snippet shows how to bring the lyrics line closer to the staff.

*% Default layout:*

```

<<
  \new Staff \new Voice = melody \relative c' {
    c4 d e f | g4 f e d | c1
  }
  \new Lyrics \lyricsto melody { aa aa aa aa aa aa aa aa aa }

  \new Staff \new Voice = melody \relative c' {
    c4 d e f | g4 f e d | c1
  }

  % Reducing the minimum space below the staff and above the lyrics:
  \new Lyrics \with {
    \override VerticalAxisGroup.nonstaff-relatedstaff-spacing =
      #'((basic-distance . 1))
  }
  \lyricsto melody { aa aa aa aa aa aa aa aa aa }
>>

```



By default, TextMark objects are aligned to so-called NonMusicalPaperColumn grobs, like the left edge of the staff or a bar line. They can be aligned to a note instead by setting the non-musical property to #f.

mark a mark b

3 mark c mark d

Stem lengths on beamed notes can be varied by overriding the beamed-lengths property of the details of the Stem. If a single value is used as an argument, the length applies to all stems. When multiple arguments are used, the first applies to eighth notes, the second to sixteenth notes and so on. The final argument also applies to all notes shorter than the note length of the final argument. Non-integer arguments may also be used.

## Alternative bar numbering

Setting the `alternativeNumberingStyle` context property, two additional methods are available for enumerating bar numbers in repeats.

```
music = \relative c' {
  \repeat volta 3 {
    c4 d e f |
    \alternative {
      \volta 1 { c4 d e f | c2 d \break }
      \volta 2 { f4 g a b | f4 g a b | f2 a | \break }
      \volta 3 { c4 d e f | c2 d } } }
  c1 \bar "|"
}
```

```
\markup "default"
{
  \music
}
```

```
\markup \typewriter "numbers"
{
  \set Score.alternativeNumberingStyle = #'numbers
  \music
}
```

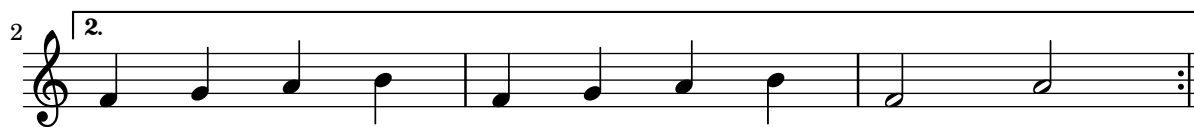
```
\markup \typewriter "numbers-with-letters"
{
  \set Score.alternativeNumberingStyle = #'numbers-with-letters
  \music
}
```

default

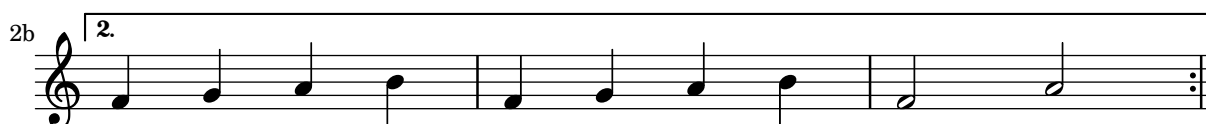
The image shows three staves of music in treble clef with a common time signature (C). The first staff has a repeat sign with a first ending bracket labeled '1.' above it. The second staff starts with a measure number '4' and has a repeat sign with a second ending bracket labeled '2.' above it. The third staff starts with a measure number '7' and has a repeat sign with a third ending bracket labeled '3.' above it. Each ending leads to a double bar line.

numbers

The image shows a single staff of music in treble clef with a common time signature (C). It has a repeat sign with a first ending bracket labeled '1.' above it, leading to a double bar line.



numbers-with-letters



## Analysis brackets above the staff

Simple horizontal analysis brackets are added below the staff by default. The following example shows a way to place them above the staff instead.

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

\relative c'' {
  \once \override HorizontalBracket.direction = #UP
  c2\startGroup
  d2\stopGroup
}
```



## Analysis brackets with labels

Text markup may be added to analysis brackets using the `text` property of the `HorizontalBracketText` grob. Adding different texts to brackets beginning at the same time requires the `\tweak` command.

Bracket text gets parenthesized after a line break.

```
\layout {
  \context {
    \Voice
```

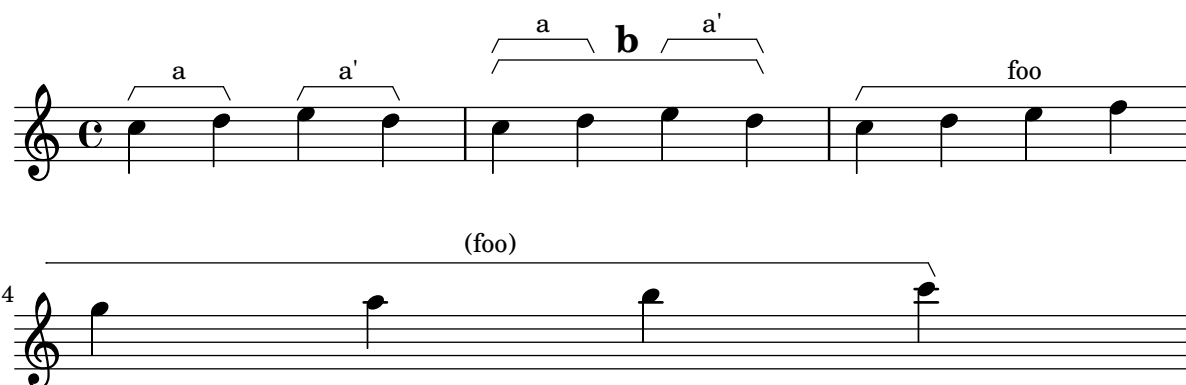


```

\consists "Horizontal_bracket_engraver"
\override HorizontalBracket.direction = #UP
}
}

{
\once\override HorizontalBracketText.text = "a"
c''\startGroup d''\stopGroup
\once\override HorizontalBracketText.text = "a'"
e''\startGroup d''\stopGroup |
c''-\tweak HorizontalBracketText.text
\markup \bold \huge "b" \startGroup
-\tweak HorizontalBracketText.text "a" \startGroup
d''\stopGroup
e''-\tweak HorizontalBracketText.text "a'" \startGroup
d''\stopGroup\stopGroup |
c''-\tweak HorizontalBracketText.text foo \startGroup
d'' e'' f'' | \break
g'' a'' b'' c'''\stopGroup
}

```



## Asymmetric slurs

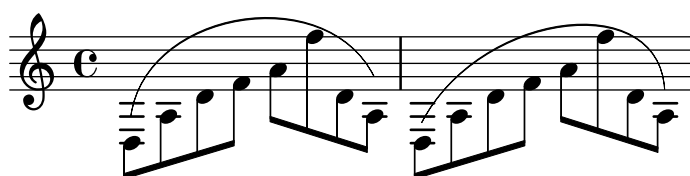
Slurs can be made asymmetric to match an asymmetric pattern of notes better.

```
slurNotes = { d,8( a' d f a f' d, a) }
```

```

\relative c' {
\stemDown
\slurUp
\slurNotes
\once \override Slur.eccentricity = #3.0
\slurNotes
}

```

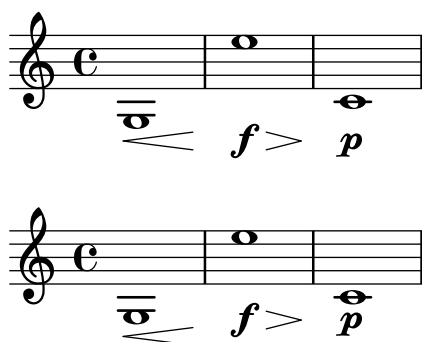


## Breaking vertical alignment of dynamics and textscripts

By default, LilyPond uses `DynamicLineSpanner` grobs to vertically align successive dynamic objects like hairpins and dynamic text. However, this is not always wanted. By inserting `\breakDynamicSpan`, which ends the alignment spanner prematurely, this vertical alignment can be avoided.

```
{ g1\< |
  e''\f\> |
  c'\p }
```

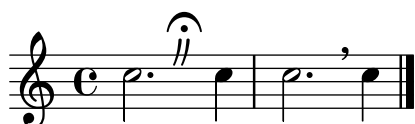
```
{ g1\< |
  e''\breakDynamicSpan\f\> |
  c'\p }
```



## Caesura (“railtracks”) with fermata

A caesura is sometimes denoted by a double “railtracks” breath mark with a fermata sign positioned above. This snippet shows an optically pleasing combination of railtracks and fermata.

```
\relative c' ' {
  c2.
  % construct the symbol
  \override BreathingSign.text = \markup {
    \override #'(direction . 1)
    \override #'(baseline-skip . 1.8)
    \dir-column {
      \translate #'(0.155 . 0)
      \center-align \musicglyph "scripts.caesura.curved"
      \center-align \musicglyph "scripts.ufermata"
    }
  }
  \breathe c4
  % set the breath mark back to normal
  \revert BreathingSign.text
  c2. \breathe c4
  \bar "|"
}
```

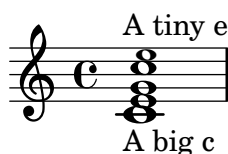


## Changing a single note's size in a chord

Individual note heads in a chord can be modified with the `\tweak` command inside a chord, by altering the `font-size` property.

Inside the chord (within the brackets `< >`), before the note to be altered, place the `\tweak` command, followed by `font-size` and define the proper size like `#-2` (a tiny note head).

```
\relative c' {
  <\tweak font-size #+2 c e g c
    \tweak font-size #-2 e>1
  ~\markup { A tiny e }_~\markup { A big c }
}
```



## Changing beam thickness and spacing

To make beams thicker or thinner alter the `Beam.beam-thickness` property. To adjust the spacing between beams alter the `Beam.length-fraction` property.

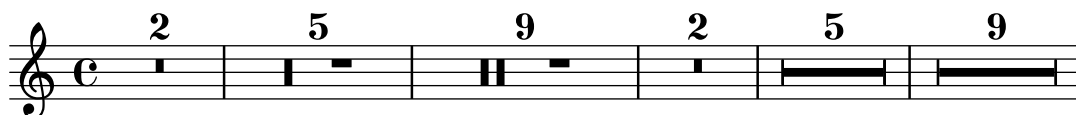
```
\relative f' {
  \time 1/8
  \override Beam.beam-thickness = #0.4
  \override Beam.length-fraction = #0.8
  c32 c c c
  \revert Beam.beam-thickness % 0.48 is default thickness
  \revert Beam.length-fraction % 1.0 is default spacing
  c32 c c c
  \override Beam.beam-thickness = #0.6
  \override Beam.length-fraction = #1.3
  c32 c c c
}
```



## 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 long, thick horizontal line is shown. This default value of ten may be changed by overriding the `expand-limit` property.

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



## Changing properties for individual grobs

The `\applyOutput` command allows the tuning of any layout object, in any context. It requires a Scheme function with three arguments.

In the example below, function `mc-squared` is executed for all `NoteHead` grobs (within the current `Voice` context) at the current time step; the function modifies the grob's stencil, using the `staff-position` property to replace some pitches with markup.

See the ‘Extending’ manual (<https://lilypond.org/doc/v2.24/Documentation/extending/running-a-function-on-all-layout-objects>) for more information.

```
#(define (mc-squared grob grob-origin context)
  (let ((sp (ly:grob-property grob 'staff-position)))
    (ly:grob-set-property!
      grob 'stencil
      (grob-interpret-markup grob
        #{ \markup \lower #0.5
          #(case sp
              ((-5) "m")
              ((-3) "c ")
              ((-2) #{ \markup \teeny \bold 2 #})
              (else "bla")) #}))))
```

```
\relative c' {
  <d f g b>2
  \applyOutput Voice.NoteHead #mc-squared
  <d f g b>2
}
```



## Changing text and spanner styles for text dynamics

The text used for *crescendos* and *decrescendos* can be changed by modifying the context properties `crescendoText` and `decrescendoText`.

The style of the spanner line can be changed by modifying the `style` property of `DynamicTextSpanner`. The default value is `dashed-line`, and other possible values include `line`, `dotted-line`, and `none`.

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



## Changing the default text font family

The default font families for text can be overridden.

```
%{
You may have to install additional fonts.

Red Hat Fedora

    dejavu-fonts-all

Debian GNU/Linux, Ubuntu

    fonts-dejavu-core
    fonts-dejavu-extra
}%}

\paper {
  %{
    run
      lilypond -dshow-available-fonts
    to show all fonts available in the process log.
  }%}
  property-defaults.fonts.serif = "DejaVu Serif"
  property-defaults.fonts.sans = "DejaVu Sans"
  property-defaults.fonts.typewriter = "DejaVu Sans Mono"
}

{
  g'''4^\markup {
    DejaVu Serif: \bold bold
                  \italic italic
                  \italic \bold { bold italic }
  }
  g4_\markup {
    \override #'(font-family . sans) {
      DejaVu Sans: \bold bold
                   \italic italic
                   \italic \bold { bold italic }
    }
  }
  g''2^\markup {
    \override #'(font-family . typewriter) {
      DejaVu Sans Mono: \bold bold
                       \italic italic
                       \italic \bold { bold italic }
    }
  }
}
```

DejaVu Serif: **bold italic bold italic**

DejaVu Sans Mono: **bold italic bold italic**

DejaVu Sans: **bold italic bold italic**

## Changing the staff size

Though the simplest way to resize staves is to use `#{set-global-staff-size size}`, an individual staff's size can be changed by scaling the properties `staff-space` and `fontSize`.

```
<<
\new Staff {
  \relative c' {
    \dynamicDown
    c8\ff c c c c c c c
  }
}
\new Staff \with {
  fontSize = #-3
  \override StaffSymbol.staff-space = #(magstep -3)
} {
  \clef bass
  c8 c c c c\ff c c c
}
>>
```

*ff*

*f*

## 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 { }
}
```



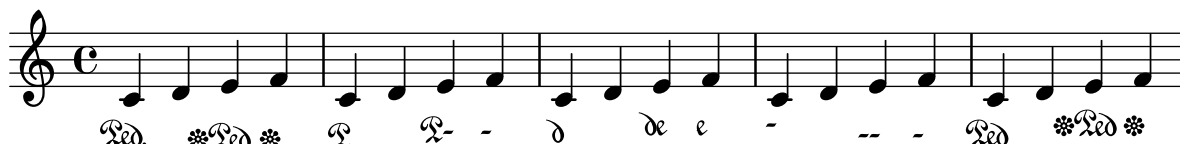
## Changing the text for sustain markings

With the `pedalSustainStrings` context property it is possible to set the text used for pedal down and up. Note that the only valid strings are those found in the list of pedal glyphs – the values shown in this snippet constitute an exhaustive list.

```
sustainNotes = { c4\sustainOn d e\sustainOff\sustainOn f\sustainOff }
```

```
\relative c' {
  \sustainNotes
  \set Staff.pedalSustainStrings = #'("P" "P-" "-")
  \sustainNotes
  \set Staff.pedalSustainStrings = #'("d" "de" "e")
  \sustainNotes
  \set Staff.pedalSustainStrings = #'("M" "M-" "-")
  \sustainNotes
  \set Staff.pedalSustainStrings = #'("Ped" "*Ped" "*")
  \sustainNotes
}
```

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



## Controlling spanner visibility after a line break

The visibility of spanners which end on the first note following a line break is controlled by the after-line-breaking callback `ly:spanner::kill-zero-spanned-time`.

For objects such as glissandos and hairpins, the default behaviour is to hide the spanner after a break; disabling the callback will allow the left-broken span to be shown.

Conversely, spanners which are usually visible, such as text spans, can be hidden by enabling the callback.

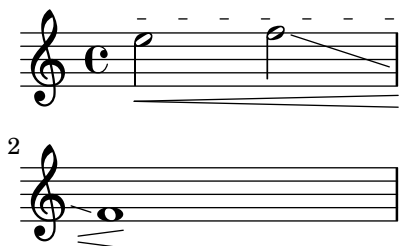
```
\paper {
  line-width = 50\mm
}
```

```
\relative c' {
  \override Hairpin.to-barline = ##f
  \override Glissando.breakable = ##t
  % show hairpin
  \override Hairpin.after-line-breaking = ##t
  % hide text span
  \override TextSpanner.after-line-breaking =
```

```

#ly:spanner::kill-zero-spanned-time
e2\<\startTextSpan
% show glissando
\override Glissando.after-line-breaking = ##t
f2\glissando
\break
f,1\!\stopTextSpan
}

```



## Controlling the appearance of tremolo slashes

Using various properties of the `StemTremolo` grob it is possible to control the appearance of tremolo slashes.

- Property `slope` sets the slope for tremolo slashes.
- Property `shape` determines whether tremolo slashes look like rectangles (value `rectangle`) or like very small beams (value `beam-like`).
- Property `style` sets both the slope and the shape depending on whether the note has flags, beams, or only a plain stem. This is in contrast to the previous two properties, which change the slope and shape unconditionally. There are two styles defined.
  - `default`: slashes for down-stem flags are longer and more sloped than slashes for up-stem flags; slashes on beamed notes have a rectangular shape and are parallel to the beam.
  - `constant`: all slashes are beam-like and have the same slope except for down-stem flags.

```

music = {
  a''4:32 a':
  e''8: \noBeam e':
  a'':[ a':]
  f':[ g':]
  d':[ d':]
}

\new Staff {
  <>^\markup "default"
  \music
}

\new Staff {
  <>^\markup \typewriter "style = #'constant"
  \override StemTremolo.style = #'constant
  \music
}

```



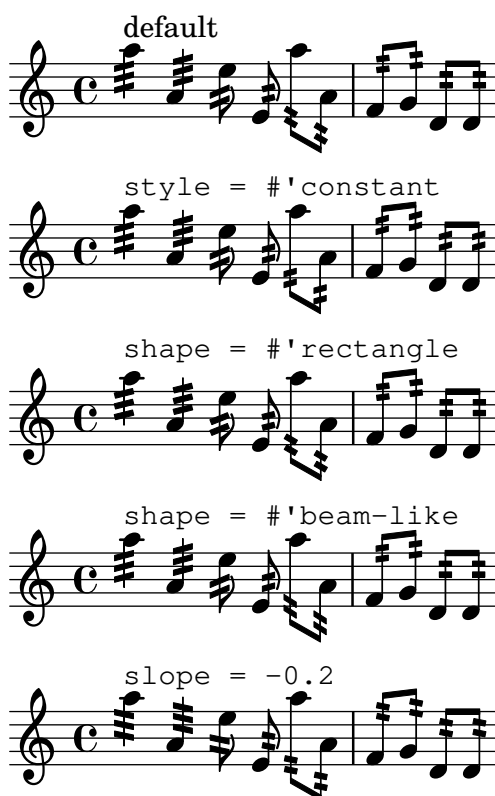
```

\new Staff {
  <>^\markup \typewriter "shape = #'rectangle"
  \override StemTremolo.shape = #'rectangle
  \music
}

\new Staff {
  <>^\markup \typewriter "shape = #'beam-like"
  \override StemTremolo.shape = #'beam-like
  \music
}

\new Staff {
  <>^\markup \typewriter "slope = -0.2"
  \override StemTremolo.slope = -0.2
  \music
}

```



## 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.

Note that for Fingering, StringNumber, and StrokeFinger grobs, if used within a chord, the vertical order is also determined by the vertical position of the associated note head, which is added to (or, depending on the direction, subtracted from) the grob's `script-priority` value. This ensures that for fingerings above a chord the lower note is associated with the lower fingering

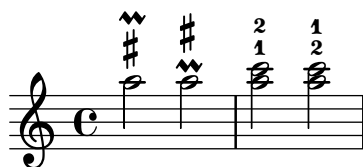
(and vice versa for the other direction); it doesn't matter whether you input the notes in the chord from top to bottom or from bottom to top.

By default, the least technical scripts are positioned closest to the note head; the rough order is articulation, flageolet, fingering, right-hand fingering, string number, fermata, bowing, and text script.

```
\relative c''' {
  \once \override TextScript.script-priority = -100
  a2^\prall^\markup { \sharp }

  \once \override Script.script-priority = -100
  a2^\prall^\markup { \sharp }

  \set fingeringOrientations = #'(up)
  <c-2 a-1>2
  <a-1 c\tweak script-priority -100 -2>2
}
```



## 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), `if-no-beam` (only print a bracket if there is no beam) or `#f` (never print a bracket). The latter is in fact equivalent to omitting the `TupletBracket` object altogether from the printed output.

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

\new Voice {
  \relative c' {
    \override Score.TextMark.non-musical = ##f
    \textMark "default" \music
    \override TupletBracket.bracket-visibility = #'if-no-beam
    \textMark \markup \typewriter "'if-no-beam" \music
    \override TupletBracket.bracket-visibility = ##t
    \textMark \markup \typewriter "#t" \music
    \override TupletBracket.bracket-visibility = ##f
    \textMark \markup \typewriter "#f" \music
    \omit TupletBracket
    \textMark \markup \typewriter "omit" \music
  }
}
```



## 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 first argument of `\after` (which is a duration) adjusts the horizontal position.

```
\relative c' {
  \after 2*2/3 \turn c2( d4) r |
  \after 4 \turn c4.( d8)
  \after 4
  {
    \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
    \once \hideNotes
    cis8\turn \noBeam
  }
  d4.( e8)
}
```



## Creating custom key signatures

LilyPond supports custom key signatures. In this example, print for D minor and D major with an extended range of shown flats.

```
\new Staff \with {
  \override StaffSymbol.line-count = #8
  \override KeySignature.flat-positions = #'((-7 . 6))
  \override KeyCancellation.flat-positions = #'((-7 . 6))
  \override KeySignature.sharp-positions = #'((-6 . 7))
  \override KeyCancellation.sharp-positions = #'((-6 . 7))

  \override Clef.stencil =
    #(lambda (grob)
      (grob-interpret-markup grob
        #{ \markup\combine
            \musicglyph "clefs.C"
            \translate #'(-3 . -2)
          }
      )
    )
}
```

```

\musicglyph "clefs.F"
#})))
clefPosition = #3
middleCPosition = #3
middleCClefPosition = #3
}

{
\key d\minor f bes, f bes, |
\key d\major fis b, fis b, |
}

```



## Creating text spanners

The `\startTextSpan` and `\stopTextSpan` commands allow the creation of text spanners as easily as pedal indications or octavations. Override some properties of the `TextSpanner` object to modify its output.

```

\paper { ragged-right = ##f }

\relative c' {
  \override TextSpanner.bound-details.left.text = #"bla"
  \override TextSpanner.bound-details.right.text = #"blu"
  a4 \startTextSpan
  b4 c
  a4 \stopTextSpan

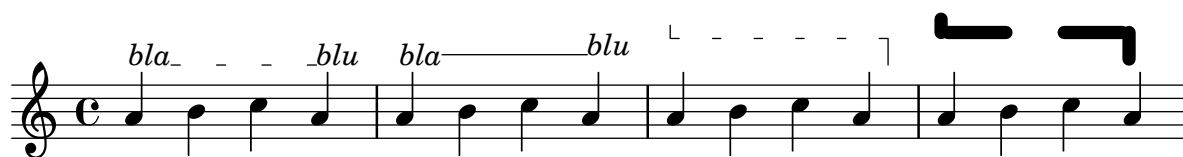
  \override TextSpanner.style = #'line
  \once \override TextSpanner.bound-details.left.stencil-align-dir-y = #CENTER
  a4 \startTextSpan
  b4 c
  a4 \stopTextSpan

  \override TextSpanner.style = #'dashed-line
  \override TextSpanner.bound-details.left.text =
    \markup { \draw-line #'(0 . 1) }
  \override TextSpanner.bound-details.right.text =
    \markup { \draw-line #'(0 . -2) }
  \once \override TextSpanner.bound-details.right.padding = #-2
  a4 \startTextSpan
  b4 c
  a4 \stopTextSpan

  \override TextSpanner.dash-period = #10
  \override TextSpanner.dash-fraction = #0.5
  \override TextSpanner.thickness = #10
  a4 \startTextSpan
  b4 c
  a4 \stopTextSpan
}

```

}



## Cross-staff chords – beaming problems workaround

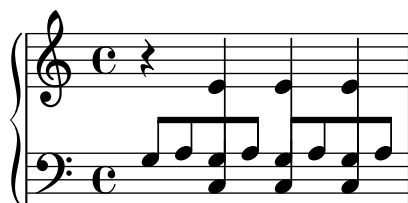
Sometimes it is better to use stems from the ‘other’ staff for creating cross-staff chords to trick LilyPond’s beam collision detector. In the following snippet, if the stems from the lower staff were used instead, it would be necessary to explicitly use

```
\override Staff.Beam.collision-voice-only = ##t
```

so that LilyPond doesn’t move the beams.

```
\new PianoStaff <<
\new Staff = up \relative c' <<
{ r4
  \override Stem.cross-staff = ##t
  \override Stem.length = #19 % this is in half-spaces,
    % so it makes stems 9.5 staffspaces long
  \override Stem.Y-offset = #-6 % stems are normally lengthened
    % upwards, so here we must lower the stem by the amount
    % equal to the lengthening - in this case (19 - 7) / 2
    % (7 is default stem length)
  e e e }
{ s4
  \change Staff = "bottom"
  \override NoteColumn.ignore-collision = ##t
  c, c c
}
>>

\new Staff = bottom \relative c' {
  \clef bass
  \voiceOne
  g8 a g a g a
}
>>
```



## Cross-staff stems

This snippet shows how to use `Span_stem_engraver` and `\crossStaff` to connect stems across staves automatically.

The stem lengths need not be specified, as the variable distance between noteheads and staves is calculated automatically. However, it is important that `\crossStaff` is applied to the correct

voice or staff (i.e., on the opposite side of where a beam is or would be positioned) to get the desired effect.

```
\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
    \voiceTwo
    % Down to lower staff
    \crossStaff { e'8 e'8 } e'4 |
  }

  \new Staff {
    \clef bass
    \voiceOne
    % Up to upper staff
    \crossStaff { <e g>4 e, g16 a8. c8 } d |
    g8 f g4 \voiceTwo g8 g g4 |
  }
>>
```



## Custodes

Custodes may be engraved in various styles.

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

\markup \with-true-dimensions % work around a cropping issue
\score {
  \new Staff \with { \consists "Custos_engraver" } \relative c' {
    \override Staff.Custos.neutral-position = #4

    \override Staff.Custos.style = #'hufnagel
    c1~"hufnagel" \break
    <d a' f'>1

    \override Staff.Custos.style = #'medicaea
```

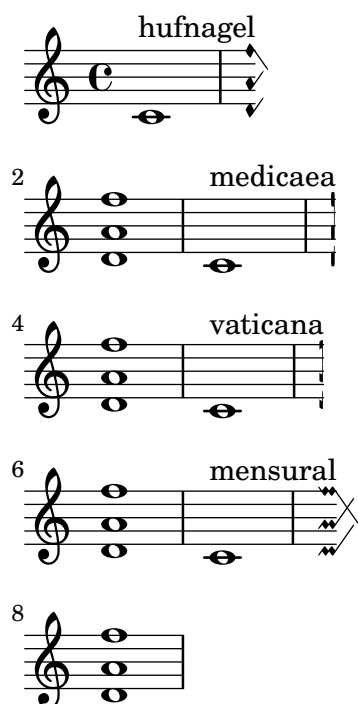
```

c1^"medicaea" \break
<d a' f'>1

\override Staff.Custos.style = #'vaticana
c1^"vaticana" \break
<d a' f'>1

\override Staff.Custos.style = #'mensural
c1^"mensural" \break
<d a' f'>1
}
}

```



## 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, and therefore can be omitted in property overrides.

```

% begin verbatim
\include "predefined-guitar-fretboards.ly"

\storePredefinedDiagram #default-fret-table \chordmode { c' }
                        #guitar-tuning
                        "x;1-1-(;3-2;3-3;3-4;1-1-);"

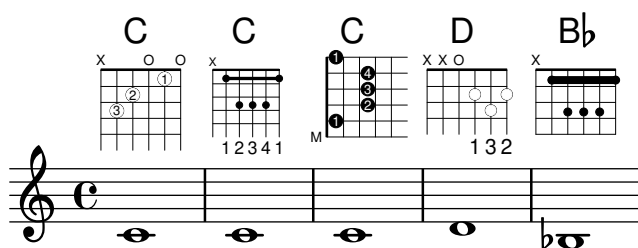
% shorthand
oo = #(define-music-function
      (grob-path value)
      (list? scheme?)
      #{ \once \override $grob-path = #value #})

```

```

<<
\new ChordNames {
  \chordmode { c1 | c | c | d | bes }
}
\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
    \oo FretBoard.size #1.0
    \oo FretBoard.fret-diagram-details.barre-type #'straight
    \oo FretBoard.fret-diagram-details.dot-color #'black
    \oo FretBoard.fret-diagram-details.finger-code #'below-string
    c'
    \oo FretBoard.fret-diagram-details.barre-type #'none
    \oo FretBoard.fret-diagram-details.number-type #'arabic
    \oo FretBoard.fret-diagram-details.orientation #'landscape
    \oo FretBoard.fret-diagram-details.mute-string "M"
    \oo FretBoard.fret-diagram-details.label-dir #LEFT
    \oo FretBoard.fret-diagram-details.dot-color #'black
    c'
    \oo FretBoard.fret-diagram-details.finger-code #'below-string
    \oo FretBoard.fret-diagram-details.dot-radius #0.35
    \oo FretBoard.fret-diagram-details.dot-position #0.5
    \oo FretBoard.fret-diagram-details.fret-count #3
    d
    \oo FretBoard.fret-diagram-details.barre-type #'straight
    \oo FretBoard.fret-diagram-details.finger-code #'none
    \oo FretBoard.fret-diagram-details.dot-radius #0.25
    \oo FretBoard.fret-diagram-details.dot-color #'black
    \oo FretBoard.fret-diagram-details.string-overhang #0.
    \oo FretBoard.fret-diagram-details.barre-thickness #2.
    bes
  }
}
\new Voice {
  c'1 | c' | c' | d' | bes
}
>>

```





## Customizing markup fret diagrams

Fret diagram properties can be modified by setting the `fret-diagram-details` property. For markup fret diagrams, overrides can be applied to the `Voice.TextScript` object or directly to the markup.

```
<<
\chords { c1 | c | c | d }

\new Voice = "mel" {
  \textLengthOn
  % Set global properties of fret diagram
  \override TextScript.size = 1.2
  \override TextScript.fret-diagram-details.finger-code = #'in-dot
  \override TextScript.fret-diagram-details.dot-color = #'white

  %% C major for guitar, no barre, using defaults
  % terse style
  c'1~\markup { \fret-diagram-terse "x;3-3;2-2;o;1-1;o;" }

  %% C major for guitar, barred on third fret
  % verbose style
  % size 1.0
  % roman fret label, finger labels below string, straight barre
  c'1~\markup {
    % standard size
    \override #'(size . 1.0) {
      \override #'(fret-diagram-details . (
        (number-type . roman-lower)
        (finger-code . in-dot)
        (barre-type . straight))) {
        \fret-diagram-verbose #'((mute 6)
          (place-fret 5 3 1)
          (place-fret 4 5 2)
          (place-fret 3 5 3)
          (place-fret 2 5 4)
          (place-fret 1 3 1)
          (barre 5 1 3))
      }
    }
  }

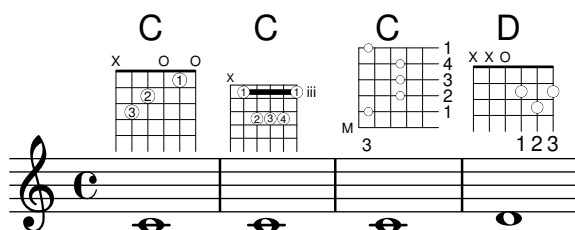
  %% C major for guitar, barred on third fret
  % verbose style
  % landscape orientation, arabic numbers, M for mute string
  % no barre, fret label down or left, small mute label font
  c'1~\markup {
    \override #'(fret-diagram-details . (
      (finger-code . below-string)
      (number-type . arabic)
      (label-dir . -1)
      (mute-string . "M")
      (orientation . landscape)
```

```

        (barre-type . none)
        (xo-font-magnification . 0.4)
        (xo-padding . 0.3))) {
\ fret-diagram-verbose #'((mute 6)
                        (place-fret 5 3 1)
                        (place-fret 4 5 2)
                        (place-fret 3 5 3)
                        (place-fret 2 5 4)
                        (place-fret 1 3 1)
                        (barre 5 1 3))
    }
}

%% simple D chord
% terse style
% larger dots, centered dots, fewer frets
% label below string
d'1~\markup {
  \override #'(fret-diagram-details . (
    (finger-code . below-string)
    (dot-radius . 0.35)
    (dot-position . 0.5)
    (fret-count . 3))) {
    \fret-diagram-terse "x;x;o;2-1;3-2;2-3;"
  }
}
}
>>

```



## Display bracket with only one staff in a system

If there is only one staff in a `ChoirStaff` or `StaffGroup` context, the bracket and the starting bar line will not be displayed by default. This can be changed by setting the `collapse-height` property to a value 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.

```

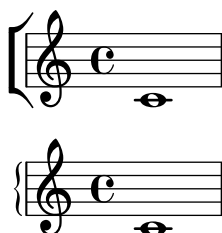
\score {
  \new StaffGroup <<
    % Must be lower than the actual number of staff lines
    \override StaffGroup.SystemStartBracket.collapse-height = 4
    \override Score.SystemStartBar.collapse-height = 4
    \new Staff {
      c'1
    }
  }
}

```

```

>>
}
\score {
  \new PianoStaff <<
    \override PianoStaff.SystemStartBrace.collapse-height = 4
    \override Score.SystemStartBar.collapse-height = 4
    \new Staff {
      c'1
    }
  >>
}

```



## Displaying grob ancestry

When working with grob callbacks, it can be helpful to understand a grob's ancestry. Most grobs have parents which influence the positioning of the grob. X- and Y-parents influence the horizontal and vertical positions for the grob, respectively. Additionally, each parent may have parents of its own.

Unfortunately, there are several aspects of a grob's ancestry that can lead to confusion:

- The types of parents a grob has may depend on context.
- For some grobs, the X- and Y-parents are the same.
- A particular *ancestor* may be related to a grob in multiple ways.
- The concept of *generations* is misleading.

For example, the System grob can be both parent (on the Y-side) and grandparent (twice on the X-side) to a VerticalAlignment grob.

The macro defined in this snippet prints (to the console) a textual representation of a grob's ancestry. For example, the call

```

{
  \once \override NoteHead.before-line-breaking = #display-ancestry
  c
}

```

generates the following output.

```

-----
NoteHead
X,Y: NoteColumn
  X: PaperColumn
    X,Y: System
  Y: VerticalAxisGroup
    X: NonMusicalPaperColumn
      X,Y: System
    Y: VerticalAlignment
      X: NonMusicalPaperColumn

```

```

      X,Y: System
    Y: System

```

As a consequence, you have to execute the code in this snippet by yourself, since the generated output file doesn't show the data we are interested in.

```

#(define (get-ancestry grob)
  (if (not (null? (ly:grob-parent grob X)))
      (list (grob::name grob)
            (get-ancestry (ly:grob-parent grob X))
            (get-ancestry (ly:grob-parent grob Y)))
      (grob::name grob)))

#(define (format-ancestry lst padding)
  (string-append
    (symbol->string (car lst)) "\n"
    (let ((X-ancestry (if (list? (cadr lst))
                          (format-ancestry (cadr lst) (+ padding 3))
                          (symbol->string (cadr lst))))
          (Y-ancestry (if (list? (caddr lst))
                          (format-ancestry (caddr lst) (+ padding 3))
                          (symbol->string (caddr lst)))))
      (if (equal? X-ancestry Y-ancestry)
          (string-append (format #f "~&")
                          (make-string padding #\space)
                          "X,Y: "
                          (if (list? (cadr lst))
                              (format-ancestry (cadr lst) (+ padding 5))
                              (symbol->string (cadr lst))))
          (string-append (format #f "~&")
                          (make-string padding #\space)
                          "X: " X-ancestry "\n"
                          (make-string padding #\space)
                          "Y: " Y-ancestry (format #f "~&"))))
    (format #f "~&")))

#(define (display-ancestry grob)
  (format (current-output-port)
    "~2&~a~2%~a~&"
    (make-string 36 #\-)
    (if (ly:grob? grob)
        (format-ancestry (get-ancestry grob) 0)
        (format #f "~a is not a grob" grob))))

\relative c' {
  \once \override NoteHead.before-line-breaking = #display-ancestry
  f4
  \once \override Accidental.before-line-breaking = #display-ancestry
  \once \override Arpeggio.before-line-breaking = #display-ancestry
  <f as c>4\arpeggio
}

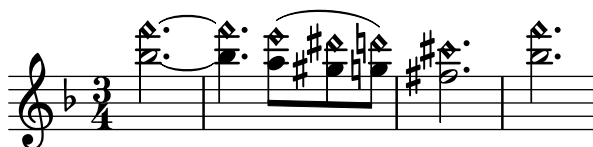
```



## Dotted harmonics

Artificial harmonics using `\harmonic` do not show dots. To override this behavior, set the context property `harmonicDots`.

```
\relative c' {} {
  \time 3/4
  \key f \major
  \set harmonicDots = ##t
  <bes f'\harmonic>2. ~
  <bes f'\harmonic>4. <a e'\harmonic>8( <gis dis'\harmonic> <g d'\harmonic>)
  <fis cis'\harmonic>2.
  <bes f'\harmonic>2.
}
```



## Drawing boxes around grobs

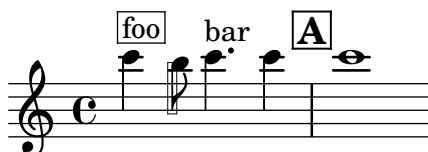
The `stencil` property can be overridden to draw a box around arbitrary grobs, either using `\override` or `\tweak`.

```
\relative c' {} {
  \once \override TextScript.stencil =
    #(make-stencil-boxer 0.1 0.3 ly:text-interface::print)
  c'4^"foo"

  \tweak Stem.stencil
    #(make-stencil-boxer 0.05 0.25 ly:stem::print)
  b8

  c4.^"bar" c4

  \override Score.RehearsalMark.stencil =
    #(make-stencil-boxer 0.15 0.3 ly:text-interface::print)
  \mark \default
  c1
}
```



## Drawing circles around various objects

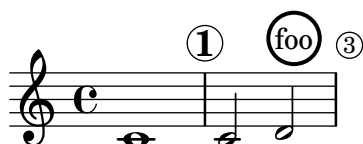
The `\circle` command draws circles around `\markup` objects. For other objects, specific tweaks may be required, as demonstrated for rehearsal marks and measure numbers.

```

\relative c' {
  c1
  \set Score.rehearsalMarkFormatter =
    #(lambda (mark context)
      (make-circle-markup (format-mark-numbers mark context)))
  \mark \default

  c2 d^\markup {
    \override #'(thickness . 3) {
      \circle foo
    }
  }
  \override Score.BarNumber.break-visibility = #all-visible
  \override Score.BarNumber.stencil =
    #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
}

```



## Dynamics spanner with custom text

Postfix functions for custom crescendo text spanners. The spanners should start on the first note of the measure. One has to use `-\mycresc`, otherwise the spanner start will rather be assigned to the next note.

```

% Two functions for (de)crescendo spanners where you can explicitly
% give the spanner text.

```

```

mycresc =
#(define-music-function (mymarkup) (markup?)
  (make-music 'CrescendoEvent
    'span-direction START
    'span-type 'text
    'span-text mymarkup))

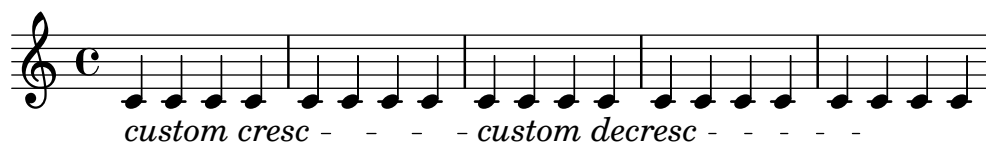
mydecresc =
#(define-music-function (mymarkup) (markup?)
  (make-music 'DecrescendoEvent
    'span-direction START
    'span-type 'text
    'span-text mymarkup))

```

```

\relative c' {
  c4-\mycresc "custom cresc" c4 c4 c4 |
  c4 c4 c4 c4 |
  c4-\mydecresc "custom decresc" c4 c4 c4 |
  c4 c4 c4 c4 |
  c4 c4\! c4 c4
}

```

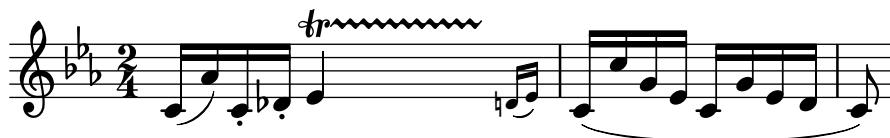


## Extending a trill spanner

For `TrillSpanner` grobs, the minimum-length property becomes effective only if the `set-spacing-rods` procedure is called explicitly.

To do this, the `springs-and-rods` property should be set to `ly:spanner::set-spacing-rods`.

```
\relative c' {
  \key c\minor
  \time 2/4
  c16( as') c,-. des-.
  \once\override TrillSpanner.minimum-length = #15
  \once\override TrillSpanner.springs-and-rods = #ly:spanner::set-spacing-rods
  \afterGrace es4\startTrillSpan { d16[(\stopTrillSpan es)] }
  c( c' g es c g' es d
  c8)
}
```



## Extending glissandi across repeats

A glissando that 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 that takes the pitch of the grace note as its argument.

Note that in polyphonic music the grace note must be matched with corresponding grace notes in all other voices.

```
repeatGliss = #(define-music-function (grace)
  (ly:pitch?)
  #{
    % the next two lines ensure the glissando is long enough
    % to be visible
    \once \override Glissando.springs-and-rods
      = #ly:spanner::set-spacing-rods
    \once \override Glissando.minimum-length = 3.5
    \once \hideNotes
    \grace $grace \glissando
  })

\score {
  \relative c'' {
    \repeat volta 3 { c4 d e f\glissando }
    \alternative {
      { g2 d }
    }
  }
}
```

```

        { \repeatGliss f g2 e }
        { \repeatGliss f e2 d }
    }
}

music = \relative c' {
  \voiceOne
  \repeat volta 2 {
    g a b c\glissando
  }
  \alternative {
    { d1 }
    { \repeatGliss c \once \omit StringNumber e1\2 }
  }
}

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

```

The image displays a musical score for a guitar. The top staff is a treble clef staff with a melody. The melody starts with a quarter note G4, followed by eighth notes A4, B4, C5, and D5, then a quarter note E5, and finally a quarter note D5. There are three first endings marked 1., 2., and 3. The bottom staff is a guitar tablature staff with five lines. The fret numbers are 0, 2, 1, 3, and 5. There is a repeat sign and a first ending marked 1. and 2.

## Fine-tuning pedal brackets

The appearance of pedal brackets may be altered in different ways.

```

\paper {
  ragged-right = ##f
}

```

```

\relative c' {
  c2\sostenutoOn c
  c2\sostenutoOff c
  c2\tweak shorten-pair #'(-7 . -2) \sostenutoOn c
}

```



```

c2\sostenutoOff c
c2\tweak edge-height #'(0 . 3) \sostenutoOn c
c2\sostenutoOff c
}

```



## Flat ties

This snippet provides a function `flared-tie` to draw a tie that consist of straight lines. It is intended as a replacement for the default tie-drawing function (i.e., a replacement argument for the `stencil` property of the `Tie` grob).

The argument of `flared-tie` is a list of coordinate pairs that specify additional points between the first and last point to span up the tie's lines. The first and last point are identical to the original tie's start and end point, respectively. The X and Y coordinate values are multiples of the bounding box length and height of the original tie (also taking care of the tie's direction); consequently, the first point has coordinates (0,0), and the last point (1,0).

The function `flare-tie` defines a shorthand for a flat tie. Further tweaking of the shape is possible by overriding `Tie.details.height-limit` or with `\shape`. It is also possible to change the custom definition on the fly.

```

#(define ((flared-tie coords) grob)
  (define (pair-to-list pair)
    (list (car pair) (cdr pair)))

  (define (normalize-coords goods x y dir)
    (map
      (lambda (coord)
        (cons (* x (car coord)) (* y dir (cdr coord))))
      goods))

  (define (my-c-p-s points thick)
    (make-connected-path-stencil points thick 1.0 1.0 #f #f))

  ;; Calling `ly:tie::print` and assigning its return value to a
  ;; variable in this outer `let` triggers LilyPond to position the
  ;; tie, allowing us to extract its extents. We only proceed,
  ;; however, if the tie doesn't get discarded (for whatever reason).
  (let ((sten (ly:tie::print grob)))
    (if (grob::is-live? grob)
      (let* ((layout (ly:grob-layout grob))
             (line-thickness (ly:output-def-lookup layout
                                                       'line-thickness))
             (thickness (ly:grob-property grob 'thickness 0.1))
             (used-thick (* line-thickness thickness))
             (dir (ly:grob-property grob 'direction))
             (xex (ly:stencil-extent sten X))
             (yex (ly:stencil-extent sten Y))
             (lenx (interval-length xex)))

```

```

        (leny (interval-length yex))
        (xtrans (car xex))
        (ytrans (if (> dir 0)(car yex) (cdr yex)))
        ;; Add last point.
        (coord-list (append coords '((1.0 . 0.0))))
        (uplist
         (map pair-to-list
              (normalize-coords coord-list lenx (* leny 2) dir))))
    (ly:stencil-translate
     (my-c-p-s uplist used-thick)
     (cons xtrans ytrans)))
'()))

% Define a default tie shape consisting of three straight lines.
#(define flare-tie
  (flared-tie '((0.1 . 0.3) (0.9 . 0.3))))

\relative c' {
  a4~ a
  \once \override Tie.stencil = #flare-tie
  a4~ a \break

  <a c e a c e a c e>~ q
  \once \override Tie.stencil = #flare-tie
  q~ q\break

  <>~\markup \small \typewriter "height-limit = 14"
  \override Tie.details.height-limit = 14
  a'4~ a
  \once \override Tie.stencil = #flare-tie
  a4~ a \break

  <>~\markup \small \typewriter "height-limit = 0.5"
  \override Tie.details.height-limit = 0.5
  a4~ a
  \once \override Tie.stencil = #flare-tie
  a4~ a \break

  \revert Tie.details.height-limit

  <>~\markup \small \typewriter
    "\shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0))"
  \shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0)) Tie
  a4~ a
  \once \override Tie.stencil = #flare-tie
  \shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0)) Tie
  a4~ a \break

  <>~\markup \small \typewriter
    "#(flared-tie '((0.2 . 2) (0.5 . -3) (0.8 . 1)))"
  \once \override Tie.stencil =
    #(flared-tie '((0.2 . 2) (0.5 . -3) (0.8 . 1)))

```

```

a4~ a
<>_markup \small \typewriter
      "#(flared-tie '((0.5 . 2)))"
\once \override Tie.stencil = #(flared-tie '((0.5 . 2)))
a'4~ a
}

```

1

2

3 height-limit = 14

4 height-limit = 0.5

5 \shape #'((0 . 0) (0 . -1) (0 . -1) (0 . 0))

6

#(flared-tie '((0.2 . 2) (0.5 . -3) (0.8 . 1)))

#(flared-tie '((0.5 . 2)))

## Force a cancellation natural before accidentals

The following example shows how to force a natural sign before an accidental.

```

\relative c' {
  \key es \major
  bes c des
  \tweak Accidental.restore-first ##t
  eis
}

```

## Forcing horizontal shift of notes

When the typesetting engine cannot cope, the following syntax can be used to override typesetting decisions. The units of measure used here are staff spaces.

```
\relative c' <<
{
  <d g>2 <d g>
}
\\
{
  <b f'>2
  \once \override NoteColumn.force-hshift = 1.7
  <b f'>2
}
>>
```



## Fret diagrams explained and developed

This snippet shows many possibilities for obtaining and tweaking fret diagrams.

```
<<
\chords {
  a1 a \bar "||" \break
  \repeat unfold 3 {
    c c c d d \bar "||" \break
  }
}

\new Voice {
  % Set global properties of fret diagram
  \override TextScript.size = 1.2
  \override TextScript.fret-diagram-details
    .finger-code = #'below-string
  \override TextScript.fret-diagram-details
    .dot-color = #'black

  % 1
  %
  % A chord for ukulele.
  a'1^\markup
    \override #'(fret-diagram-details
      . ((string-count . 4)
        (dot-color . white)
        (finger-code . in-dot)))
    \fret-diagram "4-2-2;3-1-1;2-o;1-o;"

  % 2
  %
  % A chord for ukulele, with formatting defined in definition
```

```

% string: 1.2 * size, 4 strings, 4 frets, fingerings below,
% string dot radius .35 of fret spacing, dot position 0.55 of
% fret spacing.
a'1^\markup
  \override #'(fret-diagram-details
    . ((dot-color . white)
      (open-string . "o")))
  \fret-diagram
    "s:1.2;w:4;h:3;f:2;d:0.35;p:0.55;4-2-2;3-1-1;2-o;1-o;"

%%
%% These chords will be in normal orientation
%%

% 3
%
% C major for guitar, barred on third fret: verbose style,
% roman fret label, finger labels below string, straight barre.
c'1^\markup
  % 110% of default size
  \override #'(size . 1.1)
  \override #'(fret-diagram-details
    . ((number-type . roman-lower)
      (finger-code . below-string)
      (barre-type . straight)))
  \fret-diagram-verbose #'((mute 6)
    (place-fret 5 3 1)
    (place-fret 4 5 2)
    (place-fret 3 5 3)
    (place-fret 2 5 4)
    (place-fret 1 3 1)
    (barre 5 1 3))

% 4
%
% C major for guitar, barred on third fret: double barre used
% to test barre function, verbose style.
c'1^\markup
  % 110% of default size
  \override #'(size . 1.1)
  \override #'(fret-diagram-details
    . ((number-type . arabic)
      (dot-label-font-mag . 0.9)
      (finger-code . in-dot)
      (fret-label-font-mag . 0.6)
      (fret-label-vertical-offset . 0)
      (label-dir . -1)
      (mute-string . "M")
      (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 4 2 5)
                        (barre 5 1 3))

% 5
%
% C major for guitar, with capo on third fret: verbose style.
c'1~\markup
% 110% of default size
\override #'(size . 1.1)
\override #'(fret-diagram-details
. ((number-type . roman-upper)
  (dot-label-font-mag . 0.9)
  (finger-code . none)
  (fret-label-vertical-offset . 0.5)
  (xo-font-magnification . 0.4)
  (xo-padding . 0.3)))
\ fret-diagram-verbose #'((mute 6)
                        (capo 3)
                        (open 5)
                        (place-fret 4 5 1)
                        (place-fret 3 5 2)
                        (place-fret 2 5 3)
                        (open 1))

% 6
%
% Simple D chord.
d'1~\markup
\override #'(fret-diagram-details
. ((finger-code . below-string)
  (dot-radius . 0.35)
  (string-thickness-factor . 0.3)
  (dot-position . 0.5)
  (fret-count . 3)))
\ fret-diagram-terse "x;x;o;2-1;3-2;2-3;"

% 7
%
% Simple D chord, large top fret thickness.
d'1~\markup
\override #'(fret-diagram-details
. ((finger-code . below-string)
  (dot-radius . 0.35)
  (dot-position . 0.5)
  (top-fret-thickness . 7)
  (fret-count . 3)))

```

```

\ fret-diagram-terse "x;x;o;2-1;3-2;2-3;"

%%
%% These chords will be in landscape orientation
%%
\override TextScript.fret-diagram-details
    .orientation = #'landscape

% 8
%
% C major for guitar, barred on third fret: verbose style,
% roman fret label, finger labels below string, straight
% barre.
c'1^\markup
    % 110% of default size
    \override #'(size . 1.1)
    \override #'(fret-diagram-details
        . ((number-type . roman-lower)
            (finger-code . below-string)
            (barre-type . straight)))
    \fret-diagram-verbose #'((mute 6)
        (place-fret 5 3 1)
        (place-fret 4 5 2)
        (place-fret 3 5 3)
        (place-fret 2 5 4)
        (place-fret 1 3 1)
        (barre 5 1 3))

% 9
%
% C major for guitar, barred on third fret: Double barre
% used to test barre function, verbose style.
c'1^\markup
    % 110% of default size
    \override #'(size . 1.1)
    \override #'(fret-diagram-details
        . ((number-type . arabic)
            (dot-label-font-mag . 0.9)
            (finger-code . in-dot)
            (fret-label-font-mag . 0.6)
            (fret-label-vertical-offset . 0)
            (label-dir . -1)
            (mute-string . "M")
            (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 4 2 5)
                                (barre 5 1 3))

% 10
%
% C major for guitar, with capo on third fret: verbose style.
c'1~\markup
  % 110% of default size
  \override #'(size . 1.1)
  \override #'(fret-diagram-details
    . ((number-type . roman-upper)
      (dot-label-font-mag . 0.9)
      (finger-code . none)
      (fret-label-vertical-offset . 0.5)
      (xo-font-magnification . 0.4)
      (xo-padding . 0.3)))
  \fret-diagram-verbose #'((mute 6)
    (capo 3)
    (open 5)
    (place-fret 4 5 1)
    (place-fret 3 5 2)
    (place-fret 2 5 3)
    (open 1))

% 11
%
% Simple D chord.
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;"

% 12
%
% Simple D chord, large top fret thickness.
d'1~\markup
  \override #'(fret-diagram-details
    . ((finger-code . below-string)
      (dot-radius . 0.35)
      (dot-position . 0.5)
      (top-fret-thickness . 7)
      (fret-count . 3)))
  \fret-diagram-terse "x;x;o;2-1;3-2;2-3;"

%%
%% These chords will be in opposing-landscape orientation.
%%

```



```

\override TextScript.fret-diagram-details
    .orientation = #'opposing-landscape

% 13
%
% C major for guitar, barred on third fret: verbose style,
% roman fret label, finger labels below string, straight
% barre.
c'1^\markup
    % 110% of default size
    \override #'(size . 1.1)
    \override #'(fret-diagram-details
        . ((number-type . roman-lower)
            (finger-code . below-string)
            (barre-type . straight)))
    \fret-diagram-verbose #'((mute 6)
        (place-fret 5 3 1)
        (place-fret 4 5 2)
        (place-fret 3 5 3)
        (place-fret 2 5 4)
        (place-fret 1 3 1)
        (barre 5 1 3))

% 14
%
% C major for guitar, barred on third fret: double barre
% used to test barre function, verbose style.
c'1^\markup
    % 110% of default size
    \override #'(size . 1.1)
    \override #'(fret-diagram-details
        . ((number-type . arabic)
            (dot-label-font-mag . 0.9)
            (finger-code . in-dot)
            (fret-label-font-mag . 0.6)
            (fret-label-vertical-offset . 0)
            (label-dir . -1)
            (mute-string . "M")
            (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 4 2 5)
        (barre 5 1 3))

% 15
%

```

```

% C major for guitar, with capo on third fret: verbose style.
c'1~\markup
  % 110% of default size
  \override #'(size . 1.1)
  \override #'(fret-diagram-details
    . ((number-type . roman-upper)
      (dot-label-font-mag . 0.9)
      (finger-code . none)
      (fret-label-vertical-offset . 0.5)
      (xo-font-magnification . 0.4)
      (xo-padding . 0.3)))
  \fret-diagram-verbose #'((mute 6)
    (capo 3)
    (open 5)
    (place-fret 4 5 1)
    (place-fret 3 5 2)
    (place-fret 2 5 3)
    (open 1))

% 16
%
% Simple D chord.
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;"

% 17
%
% Simple D chord, large top fret thickness.
d'1~\markup
  \override #'(fret-diagram-details
    . ((finger-code . below-string)
      (dot-radius . 0.35)
      (dot-position . 0.5)
      (top-fret-thickness . 7)
      (fret-count . 3)))
  \fret-diagram-terse "x;x;o;2-1;3-2;2-3;"
}
>>

\paper {
  ragged-right = ##t
  system-system-spacing.basic-distance = 20
}

\layout {
  \context {

```

```

\Score
\override SpacingSpanner.spacing-increment = 3
}
}

```

The image displays a musical score with three staves, each containing five measures. The first staff shows two A chords. The second staff shows three C chords and two D chords. The third staff shows three C chords and two D chords. Each chord is accompanied by a fretboard diagram showing fingerings and string muting (x). The diagrams are labeled with chord names (A, C, D) and include fingerings (1, 2, 3, 4) and muting (x) for specific strings.

## Generate special note head shapes

When a note head with a special shape cannot easily be generated with graphic markup, a drawing specification for `ly:make-stencil` can be used to generate the shape. This example gives an example for a parallelogram-shaped note head.

Unfortunately, the available commands in a drawing specification are currently not documented (<https://gitlab.com/lilypond/lilypond/-/issues/6874>); in any case, the used path sub-command has the following signature, quite similar to the `make-path-stencil` Scheme function.

```
(path thickness command-list line-cap-style line-join-style fill)
```

The commands in *command-list* resemble PostScript drawing commands but with arguments after the command name.

```

parallelogram =
#(ly:make-stencil
' (path 0.1
  (rmoveto 0 0.25
    lineto 1.2 0.75
    lineto 1.2 -0.25
    lineto 0 -0.75

```

```

        lineto 0 0.25)
      round
      round
      #t)
    (cons -0.05 1.25)
    (cons -.75 .75))

myNoteHeads = \override NoteHead.stencil = \parallelogram
normalNoteHeads = \revert NoteHead.stencil

\relative c' ' {
  \myNoteHeads
  g4 d'
  \normalNoteHeads
  <f, \tweak stencil \parallelogram b e>4 d
}

```



## Generating custom flags

The `stencil` property of the `Flag` grob can be set to a custom Scheme function to generate the glyph for the flag.

```

#(define-public (weight-flag grob)
  (let* ((stem-grob (ly:grob-parent grob X))
        (log (- (ly:grob-property stem-grob 'duration-log) 2))
        (is-up? (eqv? (ly:grob-property stem-grob 'direction) UP))
        (yext (if is-up? (cons (* log -0.8) 0) (cons 0 (* log 0.8))))
        (flag-stencil (make-filled-box-stencil '(-0.4 . 0.4) yext))
        (stroke-style (ly:grob-property grob 'stroke-style))
        (stroke-stencil (if (equal? stroke-style "grace")
                             (make-line-stencil 0.2 -0.9 -0.4 0.9 -0.4)
                             empty-stencil)))
    (ly:stencil-add flag-stencil stroke-stencil)))

% Create a flag stencil by looking up the glyph from the font
#(define (inverted-flag grob)
  (let* ((stem-grob (ly:grob-parent grob X))
        (dir (if (eqv? (ly:grob-property stem-grob 'direction) UP) "d" "u"))
        (flag (retrieve-glyph-flag "" dir "" grob))
        (line-thickness (ly:staff-symbol-line-thickness grob))
        (stem-thickness (ly:grob-property stem-grob 'thickness))
        (stem-width (* line-thickness stem-thickness))
        (stroke-style (ly:grob-property grob 'stroke-style))
        (stencil (if (null? stroke-style)
                     flag
                     (add-stroke-glyph flag stem-grob dir stroke-style "")))
        (rotated-flag (ly:stencil-rotate-absolute stencil 180 0 0)))
    (ly:stencil-translate rotated-flag (cons (- (/ stem-width 2)) 0))))

```

```

snippetexamplenotes =
{
  \autoBeamOff c'8 d'16 c'32 d'64 \acciaccatura {c'8} d'64
}

{
  \time 1/4
  <>^"Normal flags"
  \snippetexamplenotes

  <>_"Custom flag: inverted"
  \override Flag.stencil = #inverted-flag
  \snippetexamplenotes

  <>^"Custom flag: weight"
  \override Flag.stencil = #weight-flag
  \snippetexamplenotes

  <>_"Revert to normal"
  \revert Flag.stencil
  \snippetexamplenotes
}

```



## Glissandi can skip grobs

NoteColumn grobs can be skipped over by glissandi.

```

\relative c' {
  a2 \glissando
  \once \override NoteColumn.glissando-skip = ##t
  f''4 d,
}

```



## Hairpins with different line styles

Hairpins can take any style from line-interface: dashed-line, dotted-line, line, trill, or zigzag.

```

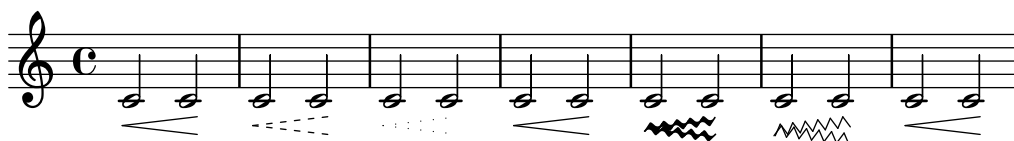
\relative c' {
  c2\< c\!
  \override Hairpin.style = #'dashed-line
  c2\< c\!
  \override Hairpin.style = #'dotted-line
  c2\< c\!
}

```

```

\override Hairpin.style = #'line
c2\< c\!
\override Hairpin.style = #'trill
c2\< c\!
\override Hairpin.style = #'zigzag
c2\< c\!
\revert Hairpin.style
c2\< c\!
}

```



## Horizontally aligning custom dynamics like “più f”

Some dynamic expressions involve additional text, like “sempre **pp**”. Since dynamics are usually centered under the note, the `\pp` would be displayed way after the note it applies to.

To correctly align the “sempre **pp**” horizontally so that it is aligned as if it were only the `\pp`, there are several approaches:

- Simply use `\once\override DynamicText.X-offset = #-9.2` before the note with the dynamics to manually shift it to the correct position. Drawback: This has to be done manually each time you use that dynamic markup...
- Add some padding (`\hspace{7.1}`) into the definition of your custom dynamic mark so that after LilyPond center-aligns it, it is already correctly aligned. Drawback: The padding really takes up that space and does not allow any other markup or dynamics to be shown in that position.
- Shift the dynamic script `\once\override ... .X-offset = ....` Drawback: `\once\override` is needed for every invocation!
- Set the dimensions of the additional text to 0 (using `\with-dimensions '(0 . 0) '(0 . 0)`). Drawback: For LilyPond, “sempre” has no extent now. This means it might put other stuff there, causing collisions (which are not detected by LilyPond’s collision detection algorithm!). There also seems to be some spacing, so it is not exactly the same alignment as without the additional text.
- Add an explicit shift directly inside the scheme function for the dynamic script.
- Set an explicit alignment inside the dynamic script. By default, this won’t have any effect, only if one sets `X-offset`! Drawback: One needs to set `DynamicText.X-offset`, which will apply to all dynamic texts! Also, it is aligned at the right edge of the additional text, not at the center of `\pp`.

```

\paper {
  ragged-right = ##f
  indent = 5\cm
}

```

```

% Solution 1: Using a simple markup with a particular halign value
% Drawback: It's a markup, not a dynamic command, so \dynamicDown
%           etc. will have no effect
semppMarkup = \markup { \halign #1.4 \italic "sempre" \dynamic "pp" }

```

```

% Solution 2: Using a dynamic script & shifting with

```

```

%           \once \override ...X-offset = ..
% Drawback: \once \override needed for every invocation
semppK =
#(make-dynamic-script
  (markup #:line
    (#:normal-text
      #:italic "sempre"
      #:dynamic "pp"))))

% Solution 3: Padding the dynamic script so the center-alignment
%           puts it at the correct position
% Drawback: the padding really reserves the space, nothing else can be there
semppT =
#(make-dynamic-script
  (markup #:line
    (#:normal-text
      #:italic "sempre"
      #:dynamic "pp"
      #:hspace 7.1)))

% Solution 4: Dynamic, setting the dimensions of the additional text to 0
% Drawback: To lilypond "sempre" has no extent, so it might put
%           other stuff there => collisions
% Drawback: Also, there seems to be some spacing, so it's not exactly the
%           same alignment as without the additional text
semppM =
#(make-dynamic-script
  (markup #:line
    (#:with-dimensions '(0 . 0) '(0 . 0)
      #:right-align
      #:normal-text
      #:italic "sempre"
      #:dynamic "pp"))))

% Solution 5: Dynamic with explicit shifting inside the scheme function
semppG =
#(make-dynamic-script
  (markup #:hspace 0
    #:translate '(-18.85 . 0)
    #:line (#:normal-text
      #:italic "sempre"
      #:dynamic "pp"))))

% Solution 6: Dynamic with explicit alignment. This has only effect
%           if one sets X-offset!
% Drawback: One needs to set DynamicText.X-offset!
% Drawback: Aligned at the right edge of the additional text,
%           not at the center of pp
semppMII =
#(make-dynamic-script
  (markup #:line (#:right-align
    #:normal-text

```

```


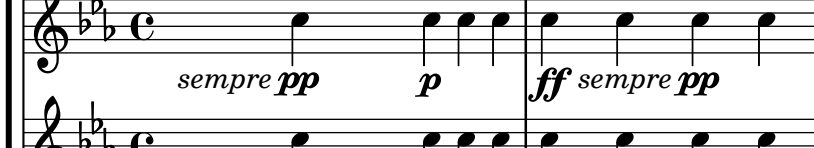


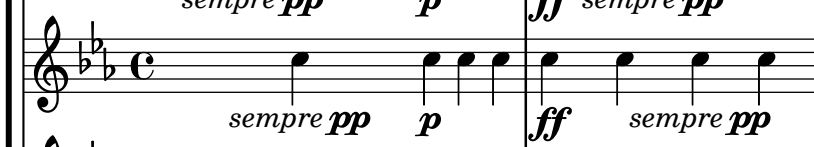
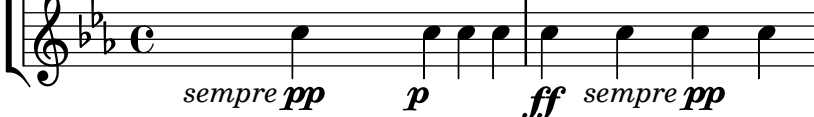
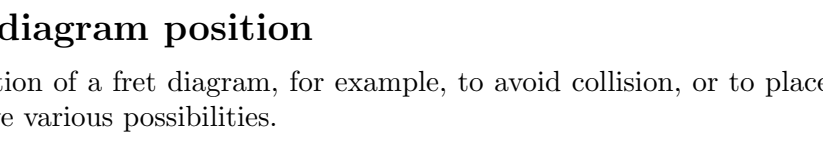
        #:italic "sempre"
        #:dynamic "pp"))))

\new StaffGroup <<
  \new Staff \with { instrumentName = "standard" }
    \relative c'' {
      \key es \major
      c4\pp c\p c c | c\ff c c\pp c
    }
  \new Staff \with {instrumentName = "normal markup" }
    \relative c'' {
      \key es \major
      c4-\semppMarkup c\p c c | c\ff c c-\semppMarkup c
    }
  \new Staff \with { instrumentName = "explicit shifting" }
    \relative c'' {
      \key es \major
      \once \override DynamicText.X-offset = #-9.2
      c4\semppK c\p c c
      c4\ff c
      \once \override DynamicText.X-offset = #-9.2
      c4\semppK c
    }
  \new Staff \with { instrumentName = "right padding" }
    \relative c'' {
      \key es \major
      c4\semppT c\p c c | c\ff c c\semppT c
    }
  \new Staff \with { instrumentName = "set dimension to zero" }
    \relative c'' {
      \key es \major
      c4\semppM c\p c c | c\ff c c\semppM c
    }
  \new Staff \with { instrumentName = "shift inside dynamics" }
    \relative c'' {
      \key es \major
      c4\semppG c\p c c | c\ff c c\semppG c
    }
  \new Staff \with { instrumentName = "alignment inside dynamics" }
    \relative c'' {
      \key es \major
      \override DynamicText.X-offset = #-1
      c4\semppMII c\p c c | c\ff c c\semppMII c
    }
}
>>

\layout { \override Staff.InstrumentName.self-alignment-X = #LEFT }

```



standard	
normal markup	
explicit shifting	
right padding	
set dimension to zero	
shift inside dynamics	
alignment inside dynamics	

## How to change fret diagram position

If you want to move the position of a fret diagram, for example, to avoid collision, or to place it between two notes, you have various possibilities.

- 1) Modify the value of the padding or extra-offset property (as shown in the first line).
- 2) You can add an invisible voice and attach the fret diagrams to the invisible notes in that voice (as shown in the second line).

If you need to move the fret according with a rhythmic position inside the bar (in the example, the third beat of the measure) the second example is better, because the fret is aligned with the third beat itself.

```

harmonies = \chordmode
{
  a8:13
  \once \override ChordNames.ChordName.extra-offset = #'(10 . 0)
  b8:13 s4. |
  s2 b2:13
}

\score {
  <<
    \new ChordNames \harmonies
    \new Staff {
      % Method 1.
      a8~\markup \fret-diagram "6-x;5-0;4-2;3-0;2-0;1-2;"
      \once \override TextScript.extra-offset = #'(10 . 0)
      b4.~\markup \fret-diagram "6-x;5-2;4-4;3-2;2-2;1-4;"
    }
  }

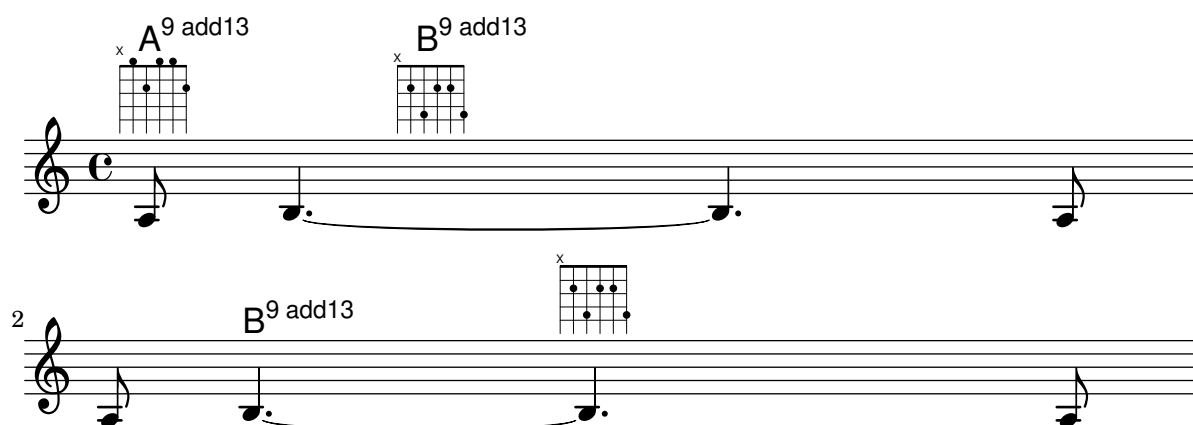
```

```

b4. a8 | \break

% Method 2.
<<
{ a8 b4.~ b4. a8 }
{ s2 s2~\markup \fret-diagram "6-x;5-2;4-4;3-2;2-2;1-4;" }
>> |
}
>>
}

```



## 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
}

```



## Keep change clefs full-sized

When a clef changes, the clef sign displayed is smaller than the initial clef. This can be overridden by setting the context property full-size-change to #t.

```

\relative c' {
  \clef "treble"
  c1
  \clef "bass"
  c1
}

```

```

\clef "treble"
c1
\override Staff.Clef.full-size-change = ##t
\clef "bass"
c1
\clef "treble"
c1
\revert Staff.Clef.full-size-change
\clef "bass"
c1
\clef "treble"
c1
}

```



## Line arrows

Arrows can be applied to text spanners and line spanners (such as glissandi).

```

\relative c' ' {
  \override TextSpanner.bound-padding = #1.0
  \override TextSpanner.style = #'line
  \override TextSpanner.bound-details.right.arrow = ##t
  \override TextSpanner.bound-details.left.text = #"fof"
  \override TextSpanner.bound-details.right.text = #"gag"
  \override TextSpanner.bound-details.right.padding = #0.6

  \override TextSpanner.bound-details.right.stencil-align-dir-y = #CENTER
  \override TextSpanner.bound-details.left.stencil-align-dir-y = #CENTER

  \override Glissando.bound-details.right.arrow = ##t
  \override Glissando.arrow-length = #0.5
  \override Glissando.arrow-width = #0.25

  a8\startTextSpan gis a4 b\glissando b,
  g'4 c\stopTextSpan c2
}

```



## Making an object invisible using the ‘transparent’ property

Setting the transparent property will cause an object to be printed in “invisible ink”: the object is not printed, but all its other behavior is retained. The object still takes up space, it takes part in collisions, and slurs, ties and beams can be attached to it.

This snippet demonstrates how to connect different voices using ties. Normally, ties only connect two notes in the same voice. By introducing a tie in a different voice, and blanking the first up-stem in that voice, the tie appears to cross voices.

```

\relative {
  \time 2/4
  <<
  {
    \once \hide Stem
    \once \override Stem.length = #8
    b'8 ~ 8\noBeam
    \once \hide Stem
    \once \override Stem.length = #8
    g8 ~ 8\noBeam
  }
  \\\
  {
    b8 g g e
  }
  >>
}

```

```

\paper {
  line-width = 40\mm
  ragged-right = ##f
}

```



## Making glissandi breakable

Normally, LilyPond refuses to automatically break a line at places where a glissando crosses a bar line. This behavior can be changed by setting the `Glissando.breakable` property to `#t`. Also setting the `after-line-breaking` property to `#t` makes the glissando line continue after the break.

The `breakable` property does not affect manual breaks inserted with commands like `\break`.

```

glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}

```

```

music = {
  \repeat unfold 16 f8 |
  f1\glissando |
  a4 r2. |
  \repeat unfold 16 f8 |
  f1\glissando \once\glissandoSkipOn |
  a2 a4 r4 |
  \repeat unfold 16 f8
}

```

```

\relative c'' {
  <>^\markup { \typewriter Glissando.breakable

```

```

        set to \typewriter "#t" }
\override Glissando.breakable = ##t
\override Glissando.after-line-breaking = ##t
\music
}

\relative c' {
  <>^\markup { \typewriter Glissando.breakable not set }
\music
}

\paper {
  line-width = 100\mm
}

```

Glissando.breakable set to #t

Glissando.breakable not set

## Manually controlling beam positions

Beam positions may be controlled manually, by overriding the positions setting of the Beam grob.

```

\relative c' {
  \time 2/4
  % from upper staff-line (position 2) to center (position 0)
  \override Beam.positions = #'(2 . 0)
  c8 c
  % from center to one above center (position 1)
  \override Beam.positions = #'(0 . 1)
  c8 c
}

```

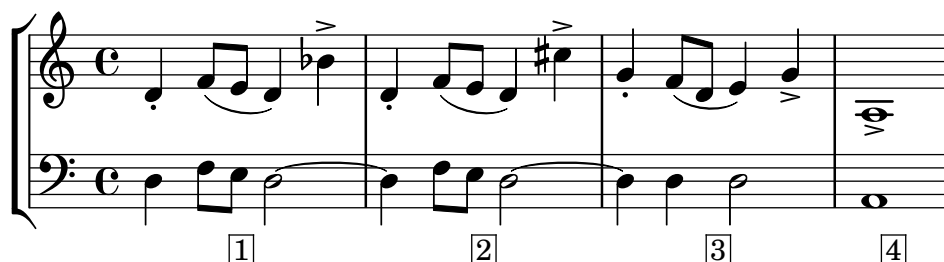
## Measure-centered bar numbers

For film scores, a common convention is to center bar numbers within their measure. This is achieved through setting the `centerBarNumbers` context property to `#t`. When this is used, the type of the bar number grobs is `CenteredBarNumber` rather than `BarNumber`.

This example demonstrates a number of settings: the centered bar numbers are boxed and placed below the staves.

```
\layout {
  \context {
    \Score
    centerBarNumbers = ##t
    barNumberVisibility = #all-bar-numbers-visible
    \override CenteredBarNumber.stencil
      = #(make-stencil-boxer 0.1 0.25 ly:text-interface::print)
    \override CenteredBarNumberLineSpanner.direction = #DOWN
  }
}

\new StaffGroup <<
  \new Staff \relative c' {
    d4-. f8( e d4) bes'-> |
    d,-. f8( e d4) cis'-> |
    g-. f8( d e4) g-> |
    a,1-> |
  }
  \new Staff \relative c {
    \clef bass
    d4 f8 e d2~ |
    4 f8 e d2~ |
    4 4 2 |
    a1 |
  }
>>
```



## Mensurstriche layout (bar lines between the staves)

*Mensurstriche*, bar lines between but not through staves, can be printed by setting `measureBarType` to `"-span|"` and using a grouping context that allows span bars, such as `StaffGroup`.

```
\layout {
  \context {
    \Staff
    measureBarType = "-span|"
  }
}
```

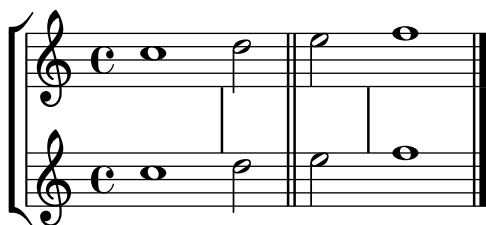
```

}

music = \fixed c'' {
  c1
  d2 \section e2
  f1 \fine
}

\new StaffGroup <<
  \new Staff \music
  \new Staff \music
>>

```



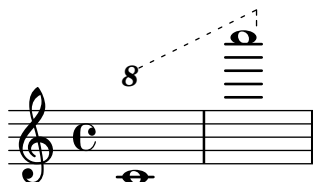
## Modifying the ottava spanner slope

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

```

\relative c'' {
  \override Staff.OttavaBracket.stencil = #ly:line-spanner::print
  \override Staff.OttavaBracket.bound-details =
    #`((left . ((Y . 0)
      (attach-dir . ,LEFT)
      (padding . 0)
      (stencil-align-dir-y . ,CENTER)))
    (right . ((Y . 5.0) ; Change the number here
      (padding . 0)
      (attach-dir . ,RIGHT)
      (text . ,(make-draw-dashed-line-markup
        (cons 0 -1.2))))))
  \override Staff.OttavaBracket.left-bound-info =
    #ly:horizontal-line-spanner::calc-left-bound-info-and-text
  \override Staff.OttavaBracket.right-bound-info =
    #ly:horizontal-line-spanner::calc-right-bound-info
  \ottava 1
  c1
  c''''1
}

```



## Moving dotted notes in polyphony

When a dotted note in the upper voice is moved to avoid a collision with a note in another voice, the default is to move the upper note to the right. This behaviour can be over-ridden by using the `prefer-dotted-right` property of `NoteCollision`.

```
\new Staff \relative c' <<
{
  f2. f4
  \override Staff.NoteCollision.prefer-dotted-right = ##f
  f2. f4
  \override Staff.NoteCollision.prefer-dotted-right = ##t
  f2. f4
}
\\
{ e4 e e e e e e e e e e }
>>
```



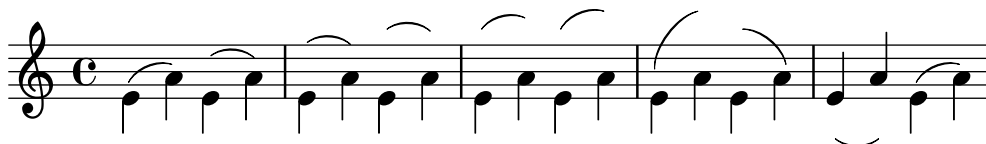
## Moving slur positions vertically

The vertical position of a slur can be adjusted using the `positions` property of `Slur`. The property has 2 parameters, the first referring to the left end of the slur and the second to the right. The values of the parameters are not used by LilyPond to make an exact movement of the slur - instead it selects what placement of the slur looks best, taking into account the parameter values. Positive values move the slur up, and are appropriate for notes with stems down. Negative values move downward slurs further down.

See also “Adjusting slur positions vertically”.

```
\relative c' {
  \stemDown
  e4( a)
  \override Slur.positions = #'(1 . 1)
  e4( a)
  \override Slur.positions = #'(2 . 2)
  e4( a)
  \override Slur.positions = #'(3 . 3)
  e4( a)
  \override Slur.positions = #'(4 . 4)
  e4( a)
  \override Slur.positions = #'(5 . 5)
  e4( a)
  \override Slur.positions = #'(0 . 5)
  e4( a)
  \override Slur.positions = #'(5 . 0)
  e4( a)
  \stemUp
  \override Slur.positions = #'(-5 . -5)
  e4( a)
  \stemDown
  \revert Slur.positions
}
```



$$\left. \begin{array}{l} e_4(a) \\ \} \end{array} \right\}$$


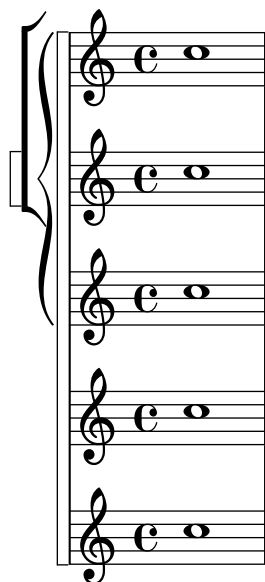
## Nesting staves

The property `systemStartDelimiterHierarchy` can be used to make more complex nested staff groups. The `systemStartDelimiterHierarchy` property of the `StaffGroup` context takes an alphabetical list of the number of staves produced. Before each staff a system start delimiter can be given. It has to be enclosed in brackets and takes as much staves as the brackets enclose. Elements in the list can be omitted, but the first bracket takes always the complete number of staves. The possibilities are `SystemStartBar`, `SystemStartBracket`, `SystemStartBrace`, and `SystemStartSquare`.

```
\new StaffGroup
\relative c' <<
  \override StaffGroup.SystemStartSquare.collapse-height = 4
  \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 }

>>
```



## Overriding articulations by type

Sometimes you may want to affect a single articulation type. Although it is always possible to use `\tweak`, it might become tedious to do so for every single sign of a whole score. The following shows how to tweak articulations with a list of custom settings. One use-case might be to create a style sheet.

```
#(define (custom-script-tweaks ls)
  (lambda (grob)
    (let* ((type (ly:event-property (ly:grob-property grob 'cause)
                                     'articulation-type))
           (tweaks (assoc-ref ls type)))
      (when tweaks
        (for-each
         (lambda (x) (ly:grob-set-property! grob (car x) (cdr x)))
         tweaks))))))
```

```
customScripts =
#(define-music-function (settings) (list?)
  #{
    \override Script.before-line-breaking =
      #(custom-script-tweaks settings)
  })
revertCustomScripts = \revert Script.before-line-breaking
```

*% Example*

*% Predefine two sets of desired tweaks.*

```
#(define my-settings-1
  '(
    (accent . ((font-size . 0)
                (color . (1 0 0))))
    (segno . ((font-size . 0)
                (color . (1 0 0))))
    (staccato . ((color . (1 0 0))
                  (padding . 0.5)))
    (staccatissimo . ((padding . 1)
                       (color . (1 0 0))))
    (tenuto . ((color . (1 0 0))
                (rotation . (45 0 0))
                (padding . 2)
                (font-size . 10)))
  ))
```

```
#(define my-settings-2
  '(
    (accent . ((font-size . 4)
                (color . (0 1 0))
                (padding . 1.5)))
    (coda . ((color . (0 1 0))
              (padding . 1)))
    (staccato . ((color . (0 1 0))))
```

```

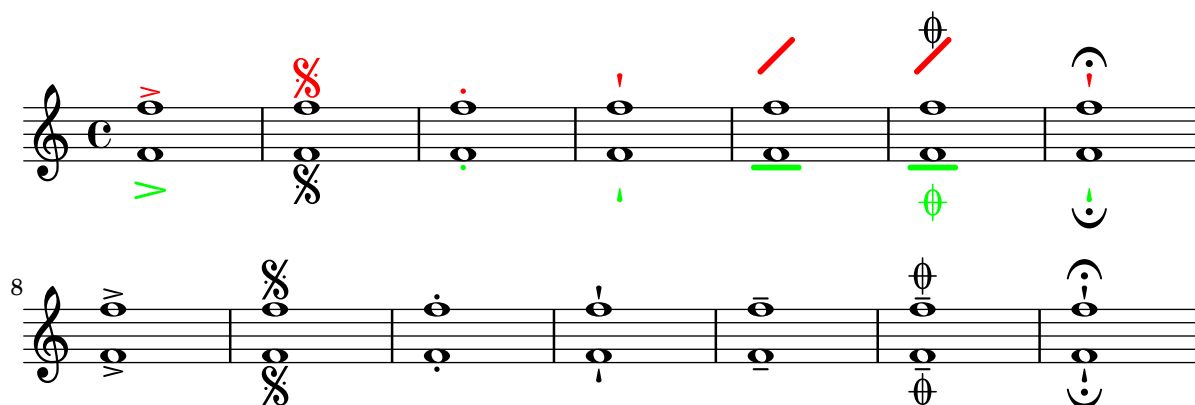
(staccatissimo . ((padding . 2)
                  (color . (0 1 0))))
(tenuto . ((color . (0 1 0))
          (font-size . 10)))
))

music = { f1-> | f\segno | f-. | f-! | f-- | f--\coda | f-!\fermata | }

block = {
  \music
  \break
  \revertCustomScripts \music
}

\new Staff <<
  \new Voice \with { \customScripts #my-settings-1 }
  \relative c'' { \voiceOne \block }
  \new Voice \with { \customScripts #my-settings-2 }
  \relative c' { \voiceTwo \block }
>>

```



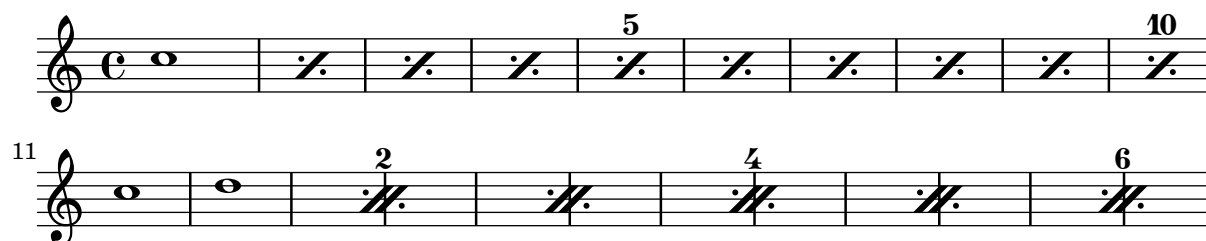
## Percent repeat count visibility

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 }
}

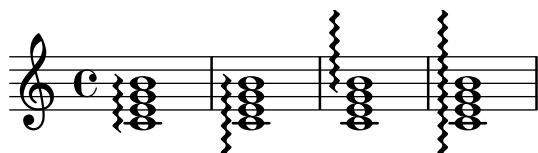
```



## Positioning arpeggios

If you need to extend or shorten an arpeggio, you can modify the upper and lower start positions independently.

```
\relative c' {
  <c e g b>1\arpeggio
  \once \override Arpeggio.positions = #'(-5 . 0)
  <c e g b>1\arpeggio
  \once \override Arpeggio.positions = #'(0 . 5)
  <c e g b>1\arpeggio
  \once \override Arpeggio.positions = #'(-5 . 5)
  <c e g b>1\arpeggio
}
```



## Positioning fingering indications precisely

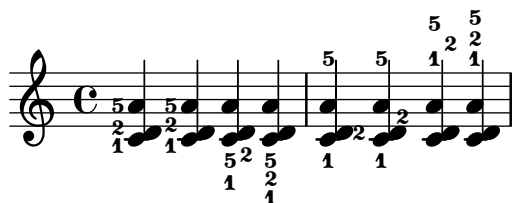
The semi-automatic positioning of fingering within a chords works usually well, but if one of the indications needs to be positioned more precisely the following tweak may be used. This is particularly useful for correcting the positioning when intervals of a second are involved.

```
\markup \with-true-dimensions % work around a cropping issue
\score {
  \relative c' {
    \set fingeringOrientations = #'(left)
    <c-1 d-2 a'-5>4
    <c-1 d-\tweak extra-offset #'(0 . 0.2)-2 a'-5>

    \set fingeringOrientations = #'(down)
    <c-1 d-2 a'-5>
    <c-\tweak extra-offset #'(0 . -1.1)-1
    d-\tweak extra-offset #'(-1.2 . -1.8)-2 a'-5> |

    \set fingeringOrientations = #'(down right up)
    <c-1 d-\tweak extra-offset #'(-0.3 . 0)-2 a'-5>4
    <c-1 d-\tweak extra-offset #'(-1 . 1.2)-2 a'-5>

    \set fingeringOrientations = #'(up)
    <c-1 d-\tweak extra-offset #'(0 . 1.1)-2
    a'-\tweak extra-offset #'(0 . 1)-5>
    <c-1 d-\tweak extra-offset #'(-1.2 . 1.5)-2
    a'-\tweak extra-offset #'(0 . 1.4)-5> |
  }
}
```



## 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.

This snippet shows how positioning of multi-measure rests can be controlled.

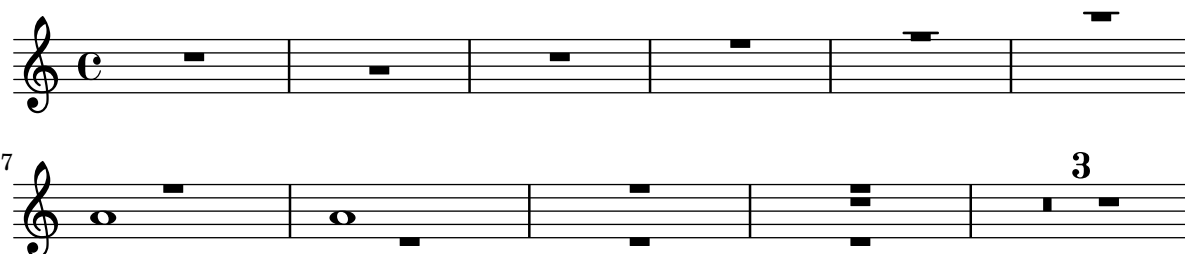
```
\relative c' ' {
  % Multi-measure rests by default are set under the fourth line.
  R1
  % They can be moved using an override.
  \override MultiMeasureRest.staff-position = -2
  R1
  \override MultiMeasureRest.staff-position = 0
  R1
  \override MultiMeasureRest.staff-position = 2
  R1
  \override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

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

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

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

```
>>
}
```



## Positioning text markups inside slurs

Text markups need to have the `outside-staff-priority` property set to `#f` 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
}
```



## Printing bar numbers inside boxes or circles

Bar numbers can also be printed inside boxes or circles.

```
\relative c' {
  % Center bar numbers except at the beginning of a staff.
  \override Score.BarNumber.self-alignment-X =
    #(break-alignment-list CENTER CENTER 0.3)

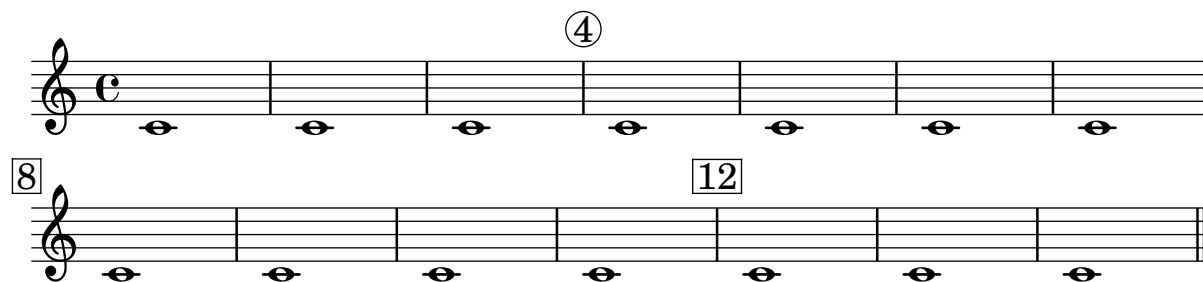
  % Prevent bar numbers at the end of a line and permit them elsewhere.
  \override Score.BarNumber.break-visibility = #end-of-line-invisible

  \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 circle round the following bar number(s).
  \override Score.BarNumber.stencil
    = #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
  \repeat unfold 7 { c1 } \break

  % 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 7 { c1 } \bar "|."
}
```



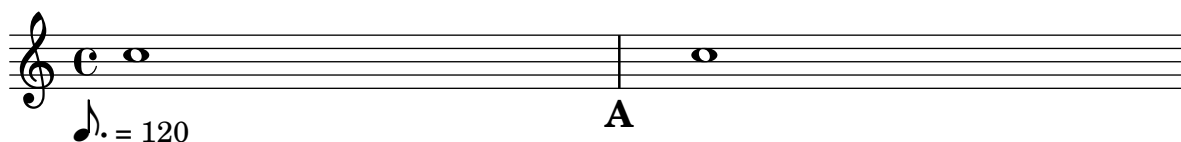
## Printing metronome and rehearsal marks below the staff

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

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

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

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



## Printing note names with and without an octave marker

The `NoteNames` context can be used to print the text value of notes. The `printOctaveNames` property turns on or off the representation of the octave of the note.

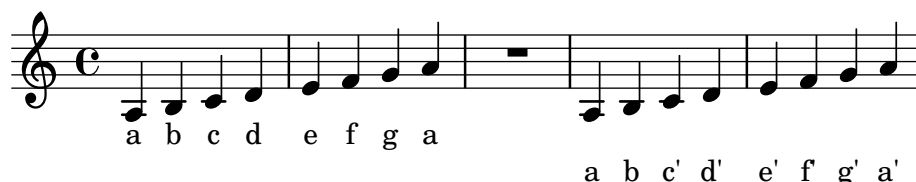
```
scale = \relative c' {
  a4 b c d
  e4 f g a
}

\new Staff {
  <<
    \scale
    \context NoteNames {
      \set printOctaveNames = ##f
      \scale
    }
  >>
  R1
  <<
```

```

\scale
\context NoteNames {
  \set printOctaveNames = ##t
  \scale
}
>>
}

```



## Printing tuplet brackets on the note head side

Whichever option you choose for controlling the tuplet bracket visibility, it will show or hide the tuplet bracket irrespectively of tuplet bracket placement (stem side or note head side). However, when placing the tuplet bracket on the note head side some authors recommend always printing the tuplet bracket. The option `visible-over-note-heads` can be used to achieve this.

```

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

\new Voice {
  \relative c' {
    \override TextScript.staff-padding = #2.5

    \time 2/4
    \override TupletBracket.visible-over-note-heads = ##t
    \override Score.TextMark.non-musical = ##f
    <>\markup "default" \music
    \override TupletBracket.bracket-visibility = #'if-no-beam
    <>\markup \typewriter "if-no-beam" \music
  }
}

```



## Proportional strict notespacing

If `strict-note-spacing` is set spacing of notes is not influenced by bars or clefs within a system. Rather, they are placed just before the note that occurs at the same time. This may cause collisions.

```

\relative c'' <<
  \override Score.SpacingSpanner.strict-note-spacing = ##t
  \set Score.proportionalNotationDuration = #1/16
  \new Staff {

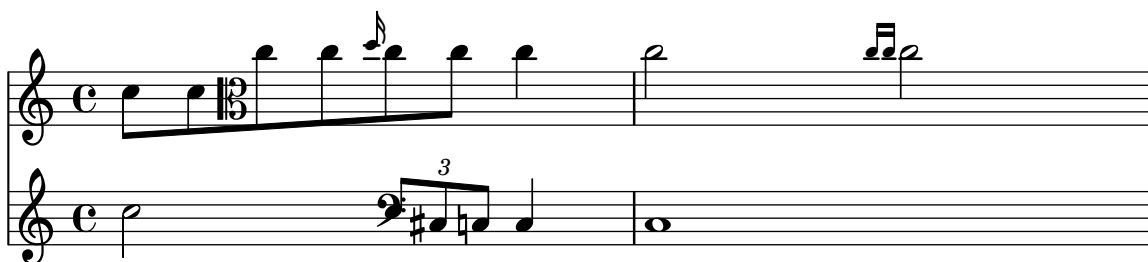
```



```

c8[ c \clef alto c c \grace { d16 } c8 c] c4
c2 \grace { c16[ c16] } c2
}
\new Staff {
  c2 \tuplet 3/2 { c8 \clef bass cis,, c } c4
  c1
}
>>

```



## Removing brace on first line of piano score

This snippet removes the first brace from a PianoStaff or a GrandStaff, together with the clefs. It may be useful when cutting and pasting the engraved image into existing music.

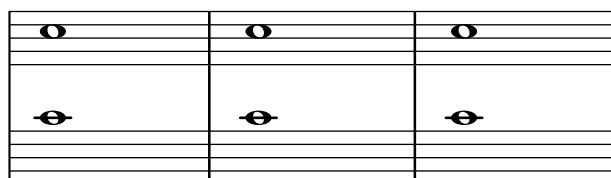
The code uses `\alterBroken` to hide the brace delimiter at the beginning.

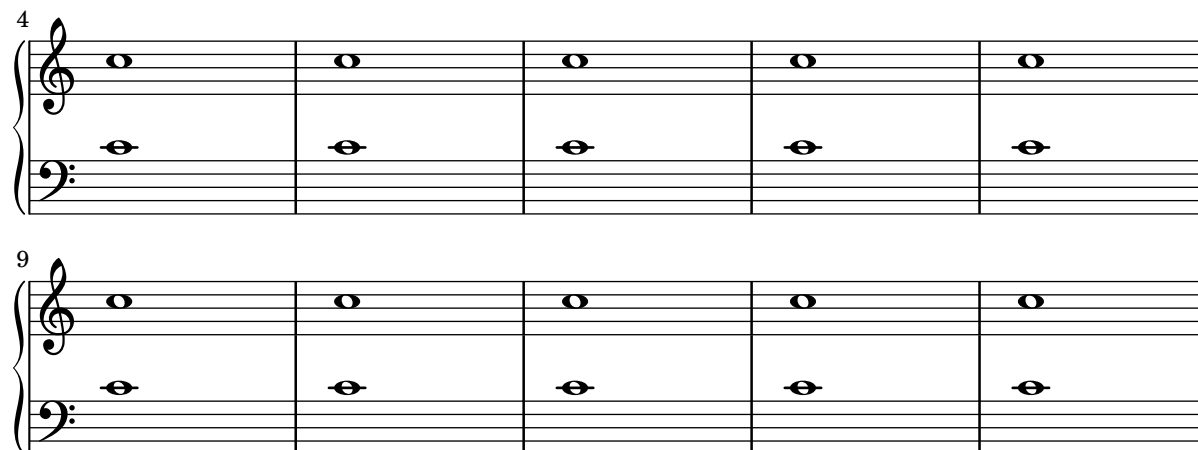
```

someMusic = {
  \once \omit Staff.Clef
  \once \omit Staff.TimeSignature
  \repeat unfold 3 c1 \break
  \repeat unfold 5 c1 \break
  \repeat unfold 5 c1
}

\score {
  \new PianoStaff
  <<
    \new Staff = "right" \relative c' \someMusic
    \new Staff = "left" \relative c' { \clef F \someMusic }
  >>
  \layout {
    indent=75\mm
    \context {
      \PianoStaff
      \alterBroken transparent #'(#t) SystemStartBrace
    }
  }
}

```

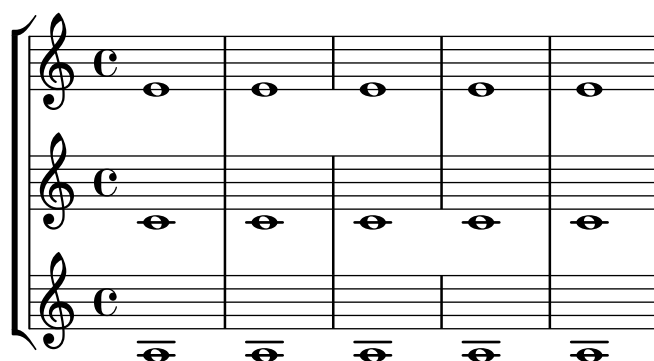




## Removing connecting bar lines on StaffGroup, PianoStaff, or GrandStaff

By default, bar lines in `StaffGroup`, `PianoStaff`, or `GrandStaff` contexts are connected between the staves, i.e., a span bar is printed. This behaviour can be overridden on a staff-by-staff basis.

```
\relative c' {
  \new StaffGroup <<
    \new Staff {
      e1 | e
      \once \override Staff.BarLine.allow-span-bar = ##f
      e1 | e | e
    }
    \new Staff {
      c1 | c | c
      \once \override Staff.BarLine.allow-span-bar = ##f
      c1 | c
    }
    \new Staff {
      a1 | a | a | a | a
    }
  >>
}
```



## Removing the first empty line

To remove the first empty staff from a score, set the `remove-first` property of the `VerticalAxisGroup` grob to `#t`. This can be done globally inside the `\layout` block or locally

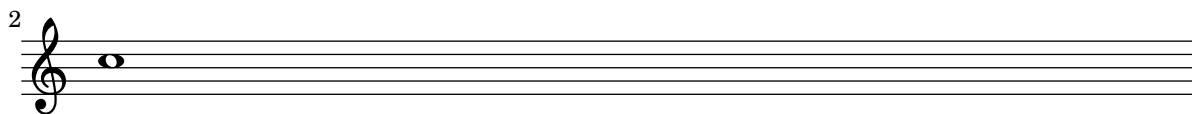
inside the specific staff that should be removed. In the latter case, you have to specify the context (`Staff` applies only to the current staff) in front of the property.

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:
    % \override VerticalAxisGroup.remove-first = ##t
  }
}
\new StaffGroup <<
  \new Staff \relative c' {
    e4 f g a \break
    c1
  }
  \new Staff {
    % To use the setting globally, comment this line,
    % uncomment the line in the \layout block above
    \override Staff.VerticalAxisGroup.remove-first = ##t
    R1 \break
    R
  }
}>>
```

```
\new StaffGroup <<
  \new Staff \relative c' {
    e4 f g a \break
    c1
  }
  \new Staff {
    R1 \break
    R
  }
}>>
```

The musical notation shows two staff groups. The first staff group consists of two staves. The top staff contains a melody of four quarter notes (e4, f, g, a) followed by a whole note (c1) on the next line. The bottom staff is empty. The second staff group also consists of two staves. The top staff contains the same melody. The bottom staff has a whole rest (R1) on the first line and a whole note (R) on the next line.



## Rest styles

Rests may be used in various styles.

```
restsA = {
  r\maxima r\longa r\breve r1 r2 r4 r8 r16 s32
  s64 s128 s256 s512 s1024 s1024
}
restsB = {
  r\maxima r\longa r\breve r1 r2 r4 r8 r16 r32
  r64 r128 r256 r512 r1024 s1024
}

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

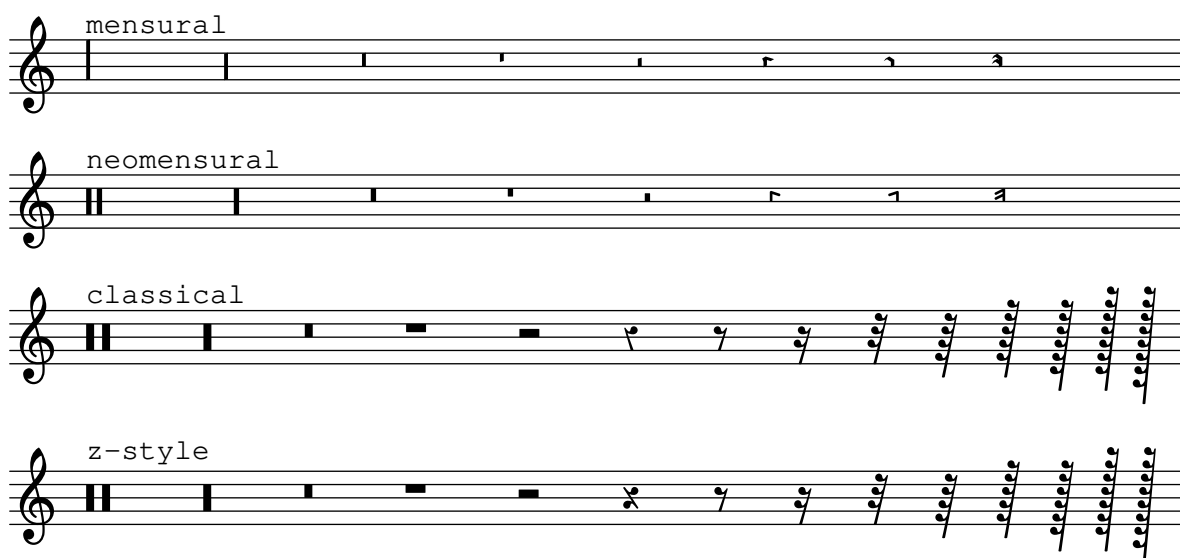
  \override Staff.Rest.style = #'mensural
  <>\markup \typewriter { mensural } \restsA \bar "" \break

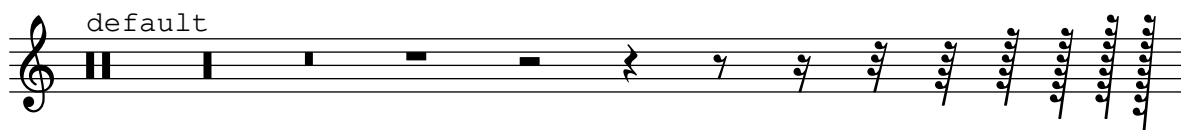
  \override Staff.Rest.style = #'neomensural
  <>\markup \typewriter { neomensural } \restsA \bar "" \break

  \override Staff.Rest.style = #'classical
  <>\markup \typewriter { classical } \restsB \bar "" \break

  \override Staff.Rest.style = #'z
  <>\markup \typewriter { z-style } \restsB \bar "" \break

  \override Staff.Rest.style = #'default
  <>\markup \typewriter { default } \restsB \bar "" \break
}
```



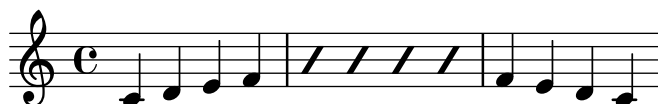


## Rhythmic slashes

In “simple” lead-sheets, sometimes no actual notes are written. Instead, only “rhythmic patterns” and chords above the measures are notated to represent the structure of a song. Such a feature can be useful while creating or transcribing the structure of a song, or when sharing lead sheets with guitarists or jazz musicians.

```
startPat = {
  \improvisationOn
  \omit Stem
}
stopPat = {
  \improvisationOff
  \undo \omit Stem
}

\new Voice \with {
  \consists Pitch_squash_engraver
} {
  c'4 d' e' f' |
  \startPat
  4 4 4 4 |
  \stopPat
  f'4 e' d' c'
}
```



## Separating key cancellations from key signature changes

By default, the accidentals used for key cancellations are placed adjacent to those for key signature changes. This behavior can be changed by overriding the `break-align-orders` property of the `BreakAlignment` grob.

The value of `break-align-orders` is a vector of length 3, with quoted lists of breakable items as elements. Each list describes the default order of prefatory matter at the end, in the middle, and at the beginning of a line, respectively. We are only interested in changing the behaviour in the middle of a line.

If you look up the definition of `break-align-orders` in LilyPond’s Internal Reference (see the `BreakAlignment` (<https://lilypond.org/doc/v2.24/Documentation/internals/breakalignment>) grob), you get the following order in the second element:

```
...
staff-bar
key-cancellation
key-signature
...
```

We want to change that, moving `key-cancellation` before `staff-bar`. To make this happen we use the `grob-transformer` function, which gives us access to the original vector as the second

argument of the lambda function, here called *orig* (we don't need the first argument, *grob*). We return a new vector, with unchanged first and last elements. For the middle element, we first remove key-cancellation from the list, then adding it again before staff-bar.

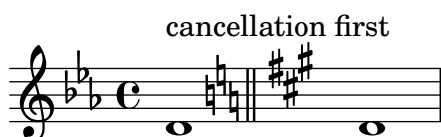
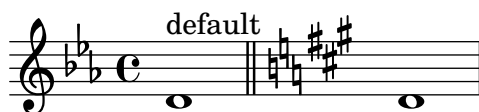
```
#(define (insert-before where what lst)
  (cond
    ((null? lst)           ; If the list is empty,
     (list what))         ; return a single-element list.
    ((eq? where (car lst)) ; If we find symbol `where`,
     (cons what lst))     ; insert `what` before curr. position.
    (else                  ; Otherwise keep building the list by
     (cons (car lst)       ; adding the current element and
            (insert-before ; recursing with the next element.
              where what (cdr lst))))))
```

```
cancellationFirst =
\override Score.BreakAlignment.break-align-orders =
#(grob-transformer
  'break-align-orders
  (lambda (grob orig)
    (let* ((middle (vector-ref orig 1))
           (middle (delq 'key-cancellation middle))
           (middle (insert-before
                        'staff-bar 'key-cancellation middle)))
      (vector
        ;; end of line
        (vector-ref orig 0)
        ;; middle of line
        middle
        ;; beginning of line
        (vector-ref orig 2)))))
```

```
music = { \key es \major d'1 \bar "||"
          \key a \major d'1 }
```

```
{ <>^\markup "default"
  \music }
```

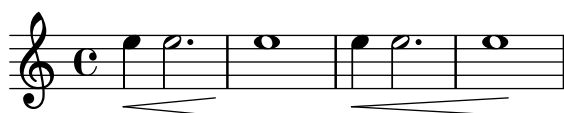
```
{ <>^\markup "cancellation first"
  \cancellationFirst
  \music }
```



## Setting hairpin behavior at bar lines

If the note which ends a hairpin falls on a downbeat, the hairpin stops at the bar line immediately preceding. This behavior can be controlled by overriding the `to-barline` property.

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



## Setting system separators

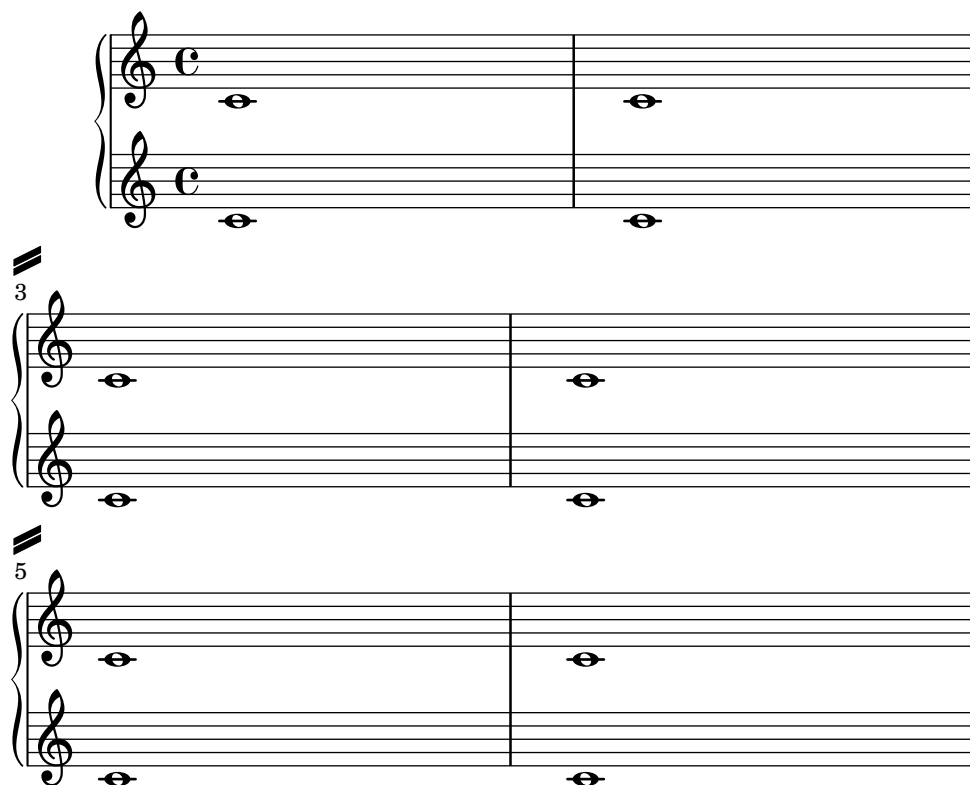
System separators can be inserted between systems. Any markup can be used, but `\slashSeparator` has been provided as a sensible default.

```
#{set-default-paper-size "a5")

\paper {
  system-separator-markup = \slashSeparator
  tagline = ##f
}

notes = \relative c' {
  c1 | c \break
  c1 | c \break
  c1 | c
}

\book {
  \score {
    \new GrandStaff <<
      \new Staff \notes
      \new Staff \notes
    >>
  }
}
```



## Shape individual ties in chords

To shape individual ties in chords use the method demonstrated below.

```
{
  \textMark "Chords can be tied note by note."
  <c'~ e'~ g'~ c''~>2 q
}

{
  \textMark \markup \override #'(baseline-skip . 3) \wordwrap {
    Modifying those ties with \typewriter "\\shape" does not succeed,
    because \typewriter TieColumn positions them on its own behalf,
    ignoring \typewriter "\\shape" input more or less. You may
    circumvent this by setting \typewriter positioning-done to
    \typewriter "#t" -- alas, \typewriter positioning-done is an
    internal property, and setting it to \typewriter "#t" means: all
    positioning is done, don't do anything further. The next example
    demonstrates a case where the positioning is not finished: all tie
    directions are down, and the thickness is not accurate.
  }
  <c'~ e'~ g'~ c''~>2
  \once \override TieColumn.positioning-done = ##t
  q
}

{
  \textMark "To fix that, enter ties with explicit direction modifiers."
  <c'_~ e'_~ g'_~ c''^~>2
  \once \override TieColumn.positioning-done = ##t
}
```



```

q
}

{
  \textMark \markup {
    Now you can use \typewriter "\\shape" for each tie as usual. }
  <c'-\shape #'((0 . 0) (0 . -10) (0 . -10) (0 . 0)) _~
  e'-\shape #'((0 . 0) (0 . -5) (0 . -5) (0 . 0)) _~
  g'-\shape #'((0 . 0) (0 . -2) (0 . -2) (0 . 0)) _~
  c''-\shape #'((0 . 0) (0 . 5) (0 . 5) (0 . 0)) ^~
  >2
  \once \override TieColumn.positioning-done = ##t
  q
}

{
  \textMark "This also works at line breaks."
  <c'-\shape #'(((0 . 0) (0 . -10) (0 . -10) (0 . 0))
    ((0 . 0) (0 . -10) (0 . -10) (0 . 0))) _~
  e'-\shape #'(((0 . 0) (0 . -5) (0 . -5) (0 . 0))
    ((0 . 0) (0 . -5) (0 . -5) (0 . 0))) _~
  g'-\shape #'(((0 . 0) (0 . -2) (0 . -2) (0 . 0))
    ((0 . 0) (0 . -2) (0 . -2) (0 . 0))) _~
  c''-\shape #'(((0 . 0) (0 . 5) (0 . 5) (0 . 0))
    ((0 . 0) (0 . 5) (0 . 5) (0 . 0))) ^~
  >2
  \break
  \once \override TieColumn.positioning-done = ##t
  q
}

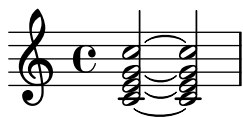
{
  \textMark \markup {
    It also works with the \typewriter tieWaitForNote property. }
  \set tieWaitForNote = ##t
  c'4-\shape #'((0 . 0) (0 . -10) (0 . -10) (0 . 0)) _~
  e'-\shape #'((0 . 0) (0 . -5) (0 . -5) (0 . 0)) _~
  g'-\shape #'((0 . 0) (0 . -2) (0 . -2) (0 . 0)) _~
  c''-\shape #'((0 . 0) (0 . 5) (0 . 5) (0 . 0)) ^~
  \once \override TieColumn.positioning-done = ##t
  <c' e' g' c''>1
}

\layout {
  indent = 0
  \context {
    \Score
    \override TextMark.padding = #4
    \override TextMark.break-align-symbols = #'(left-edge)
  }
}

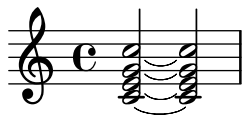
```

```
\paper {
  score-system-spacing.padding = 3
}
```

Chords can be tied note by note.



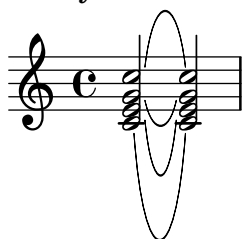
Modifying those ties with `\shape` does not succeed, because `TieColumn` positions them on its own behalf, ignoring `\shape` input more or less. You may circumvent this by setting `positioning-done` to `#t` – alas, `positioning-done` is an internal property, and setting it to `#t` means: all positioning is done, don't do anything further. The next example demonstrates a case where the positioning is not finished: all tie directions are down, and the thickness is not accurate.



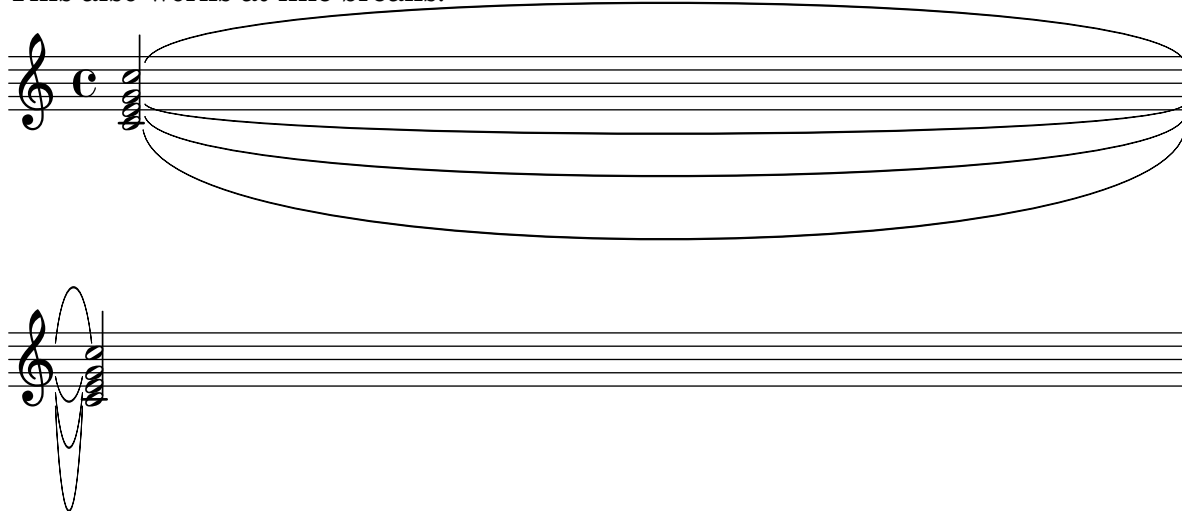
To fix that, enter ties with explicit direction modifiers.



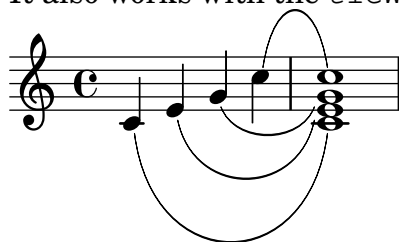
Now you can use `\shape` for each tie as usual.



This also works at line breaks.



It also works with the `tieWaitForNote` property.



## Showing the same articulation above and below a note or chord

By default, LilyPond does not allow the same articulation (an accent, a fermata, a flageolet, etc.) to be displayed above and below a note. For example, `c4_\fermata^\fermata` only shows a fermata below. The fermata above gets simply ignored.

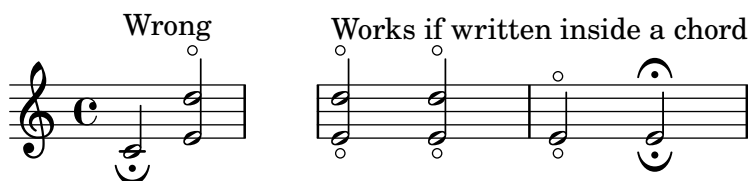
However, one can stick scripts (just like fingerings) inside a chord, which means it is possible to have as many articulations as desired. This approach has the advantage that it ignores the stem and positions the articulation relative to the note head. This can be seen in the case of the flageolets in the snippet. To mimic the behaviour of scripts outside a chord, `add-stem-support` would be required.

The solution is thus to write the note as a chord and add the articulations inside of `<...>`, using the direction modifiers `^` and `_` as appropriate.

```
\relative c' {
  <>^"Wrong"
  c2_\fermata^\fermata % The second fermata is ignored!
  <e d'>2^\flageolet_\flageolet

  \stopStaff s1 \startStaff

  <>^"Works if written inside a chord"
  <e_\flageolet d'^\flageolet>2
  <e_\flageolet d'^\flageolet>2
  <e_\flageolet^\flageolet>2
  <e_\fermata^\fermata>2
}
```



## String number extender lines

Make an extender line for string number indications, showing that a series of notes is supposed to be played all on the same string.

```
stringNumberSpanner =
  #(define-music-function (StringNumber) (string?)
    #{
      \override TextSpanner.style = #'solid
      \override TextSpanner.font-size = #-5
      \override TextSpanner.bound-details.left.stencil-align-dir-y = #CENTER
      \override TextSpanner.bound-details.left.text =
```

```

\markup { \circle \number $StringNumber }
#})

\relative c {
  \clef "treble_8"
  \textSpannerDown
  \stringNumberSpanner "5" a8\startTextSpan b c d
  e f\stopTextSpan \stringNumberSpanner "4" g\startTextSpan a |
  bes4 a g2\stopTextSpan
}

```



## Suppressing warnings for clashing note columns

If notes from two voices with stems in the same direction are placed at the same position, and both voices have no shift or the same shift specified, the error message “warning: ignoring too many clashing note columns” appears when compiling the LilyPond file. This message can be suppressed by setting the `ignore-collision` property of the `NoteColumn` object to `#t`. Please note that this does not just suppress warnings but stops LilyPond trying to resolve collisions at all and so may have unintended results unless used with care.

```
ignore = \override NoteColumn.ignore-collision = ##t
```

```

\relative c' {
  \new Staff <<
    \new Voice { \ignore \stemDown f2 g }
    \new Voice { c2 \stemDown c, }
  >>
}

```



## Time signature in brackets

The time signature can be enclosed within brackets.

```

\relative c'' {
  \override Staff.TimeSignature.stencil = #(\lambda (grob)
    (bracketify-stencil (ly:time-signature::print grob) Y 0.1 0.2 0.1))
  \time 2/4
  a4 b8 c
}

```



## Time signature in parentheses

The time signature can be enclosed within parentheses.

```
\relative c' ' {
  \override Staff.TimeSignature.stencil = #(lambda (grob)
    (parenthesize-stencil (ly:time-signature::print grob) 0.1 0.4 0.4 0.1))
  \time 2/4
  a4 b8 c
}
```



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

Sometimes, a time signature should not print the whole fraction (for example, 7/4), but only the numerator (digit 7 in this case). This can be easily done by using `\override Staff.TimeSignature.style = #'single-number` to change the style permanently. By using `\revert Staff.TimeSignature.style`, this setting can be reversed. To apply the single-number 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-number
  \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-number style only for the next time signature
  \once \override Staff.TimeSignature.style = #'single-number
  \time 5/4
  c4 c c c c
  \time 2/4
  c4 c
}
```



## Tuplet bracket and change staff

This snippet shows how to set a tuplet starting in a lower staff and finishing in the upper one.

```
aigues = \relative c' {
  \time 6/8
  s4.
```

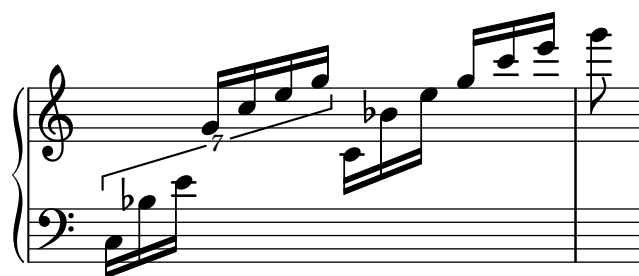
```

\stemDown
c16[ bes' e]
\stemUp
g c e
\stemDown
g8
}

basses = \relative c {
  \time 3/4
  \clef F
  \tweak positions #'(4.5 . 9.5)
  \tweak edge-height #'(1 . -1)
  \tuplet 7/6 {
    c16[ bes' e]
    \change Staff = md
    \stemUp
    g[ c e g]
  }
  s4.s8
}

\new PianoStaff
\with { \omit TimeSignature }
<<
  \new Staff = md \aigues
  \new Staff = mg \basses
>>

```



## Tweaking clef properties

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

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

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

```
{
```

```

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

% The following clef changes do not preserve
% the normal relationship between notes, key signatures
% and clefs.
\set Staff.clefGlyph = "clefs.F"
\set Staff.clefPosition = 2
c'1
\set Staff.clefGlyph = "clefs.G"
c'1
\set Staff.clefGlyph = "clefs.C"
c'1
\set Staff.clefTransposition = 7
c'1
\set Staff.clefTransposition = 0
\set Staff.clefPosition = 0
c'1

% Return to the normal clef.
\set Staff.middleCPosition = 0
c'1

```

}



## Tweaking grace layout within music

The appearance of grace expressions can be changed by using the functions `add-grace-property` and `remove-grace-property`.

The following example undefines the `direction` property of Stem grobs 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
    }
  }
}
```



## Using alternative flag styles

Alternative shapes for flags on eighth and shorter notes can be displayed by overriding the `stencil` property of Flag. LilyPond provides the following functions: `modern-straight-flag`, `old-straight-flag`, and `flat-flag`. Use `\revert` to restore the default shape.

To get stacked (i.e., vertically more compact) flags, call the command `\flagStyleStacked`, which can be reset with `\flagStyleDefault`.

Overriding the Flag stencil does not change how flag elements are positioned vertically. This is especially noticeable for flat flags: LilyPond doesn't dynamically adjust the vertical gaps between flag elements in the same way as it does for beams. A possible solution to harmonize the appearance is to replace flat flags with half beams, as shown in the second staff; however, this can't be done automatically. In the code of this snippet, such half beams are entered with `@` as a prefix, for example `@c8`.

Be aware that half beams are *not* Flag grobs. This means in particular that modifying Flag properties won't have any effect on them (you have to use Beam properties instead), and properties for their associated Stem grob will also behave beam-like.

```
"@" =
#(define-music-function (music) (ly:music?)
  #{ \set stemLeftBeamCount = 0 $music [] #})
```



```

testnotes = {
  \autoBeamOff
  c8 d16 e''32 f64 \acciaccatura { g,,,8 } a128 b
}

\relative c' {
  \override TextScript.staff-padding = 6
  \time 1/4
  <>^"default" \testnotes
  \override Flag.stencil = #modern-straight-flag
  <>_"modern straight" \testnotes
  \override Flag.stencil = #old-straight-flag
  <>^"old straight" \testnotes
  \override Flag.stencil = #flat-flag
  <>_"flat" \testnotes
  \revert Flag.stencil

  \flagStyleStacked
  <>^"stacked" \testnotes
  \flagStyleDefault
  <>_"default" \testnotes
}

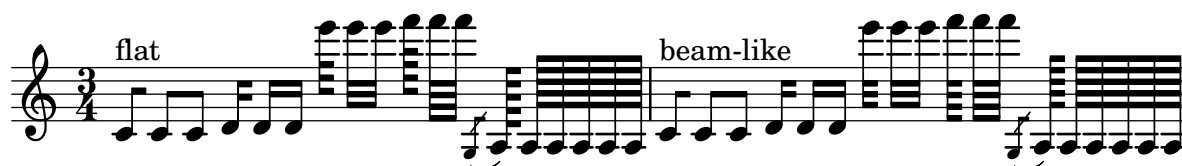
\relative c' {
  \time 3/4
  \override Flag.stencil = #flat-flag

  <>^"flat" c8 c[ c] d16 d[ d] e''32 e[ e] f64 f[ f]
  \acciaccatura { g,,,8 } a128 a[ a a a a]
  <>^"beam-like" @c8 c[ c] @d16 d[ d] @e''32 e[ e] @f64 f[ f]
  \acciaccatura { g,,,8 } @a128 a[ a a a a]
}

\layout {
  indent = 0
  \context {
    \Score
    \override NonMusicalPaperColumn.line-break-permission = ##f
  }
}

```





## Using `ly:grob-object` to access grobs with `\tweak`

Some grobs can be accessed “laterally” from within another grob’s callback. These are usually listed as “layout objects” in the “Internal properties” section of a grob interface. The function `ly:grob-object` is used to access these grobs.

Demonstrated below are some ways of accessing grobs from within a `NoteHead` callback, but the technique is not limited to `NoteHeads`. However, the `NoteHead` callback is particularly important, since it is the implicit callback used by the `\tweak` command.

The console output of the example function below (`display-grobs`) is as follows.

```
-----
#<Grob Accidental >
()
#<Grob Stem >
```

It is probably not that useful, but it demonstrates that the grobs are indeed being accessed.

```
#(define (notehead-get-accidental notehead)
  ;; notehead is grob
  (ly:grob-object notehead 'accidental-grob))

#(define (notehead-get-arpeggio notehead)
  ;; notehead is grob
  (let ((notecolumn (notehead-get-notecolumn notehead)))
    (ly:grob-object notecolumn 'arpeggio)))

#(define (notehead-get-notecolumn notehead)
  ;; notehead is grob
  (ly:grob-parent notehead X))

#(define (notehead-get-stem notehead)
  ;; notehead is grob
  (let ((notecolumn (notehead-get-notecolumn notehead)))
    (ly:grob-object notecolumn 'stem)))

#(define (display-grobs notehead)
  ;; notehead is grob
  (let ((accidental (notehead-get-accidental notehead))
        (arpeggio (notehead-get-arpeggio notehead))
        (stem (notehead-get-stem notehead)))
    (format (current-error-port) "~2&~a\n" (make-string 20 #\ -))
    (for-each
      (lambda (x) (format (current-error-port) "~a\n" x))
      (list accidental arpeggio stem))))

\relative c' {
  %% display grobs for each note head:
  \%override NoteHead.before-line-breaking = #display-grobs
  <c
```

```

%% or just for one:
\tweak before-line-breaking #display-grobs
es
g>1\arpeggio
}

```



## Using the `\tweak` command to tweak individual grobs

With the `\tweak` command, every grob can be tuned directly. Here are some examples of available tweaks.

```

\relative c' {
  \time 2/4
  \set fingeringOrientations = #'(right)
  <
    \tweak font-size #3 c
    \tweak color #red d-\tweak font-size #8 -4
    \tweak style #'cross g
    \tweak duration-log #2 a
  >2
}

```



## Vertically aligned dynamics and textscripts

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

The same idea, together with `\textLengthOn`, is used to align the text scripts along their baseline.

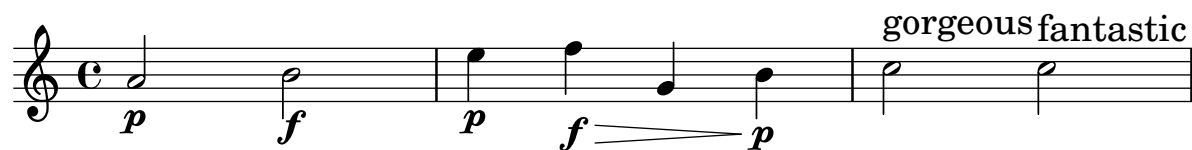
```

music = \relative c' {
  a'2\p b\f
  e4\p f\f\> g, b\p
  c2^\markup { \huge gorgeous } c^\markup { \huge fantastic }
}

{
  \music
  \break
  \override DynamicLineSpanner.staff-padding = 3
  \textLengthOn
  \override TextScript.staff-padding = 1
  \music
}

\paper { tagline = ##f }

```

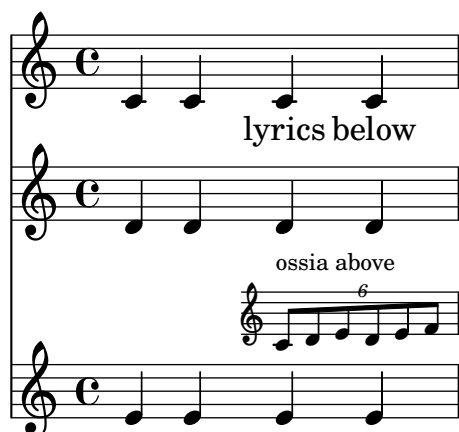


## Vertically aligning ossia and lyrics

This snippet demonstrates the use of the context properties `alignBelowContext` and `alignAboveContext` to control the positioning of lyrics and ossia.

```
\relative c' <<
  \new Staff = "1" { c4 c c c }
  \new Staff = "2" { d4 d d d }
  \new Staff = "3" { e4 e e e }

  { \skip 2
    <<
      \lyrics {
        \set alignBelowContext = "1"
        lyrics4 below
      }
      \new Staff \with {
        alignAboveContext = "3"
        fontSize = -2
        \override StaffSymbol.staff-space = #(magstep -2)
        \remove "Time_signature_engraver"
        \override VerticalAxisGroup.staff-staff-spacing =
          #'((minimum-distance . 0)
             (basic-distance . 0)
             (padding . 1))
      } {
        \tuplet 6/4 {
          \override TextScript.padding = 2
          c8[~"ossia above" d e d e f]
        }
      }
    }
  }
  >>
}
```



## Vertically aligning stanza numbers of different staves

It can happen that stanza numbers don't align vertically if the verses are attached to different staves. To fix that, override the `self-alignment-X` property of the `LyricText` grob.

```
\markup { default behavior }
```

```
<<
  \new Staff { b b b b }
  \lyrics {
    \set stanza = "3."
    a a a a
  }
```

```

  \new Staff { b b b b }
  \lyrics {
    \set stanza = "1."
    aaaaaaaaaa a a a
  }
  \lyrics {
    \set stanza = "2."
    a a a a
  }
>>
```

```
\markup \vspace #1
\markup {
  using \typewriter "self-alignment-X = #LEFT" }
```

```
<<
  \new Staff { b b b b }
  \new Lyrics \lyricmode {
    \set stanza = "3."
    a a a a
  }

  \new Staff { b b b b }
  \new Lyrics \lyricmode {
    \set stanza = "1."
    \once \override LyricText.self-alignment-X = #LEFT
```

```

      aaaaaaaaa a a a
    }
    \new Lyrics \lyricmode {
      \set stanza = "2."
      a a a a
    }
  >>

```

default behavior



using self-alignment-X = #LEFT



## Vertically centering paired figured bass extenders

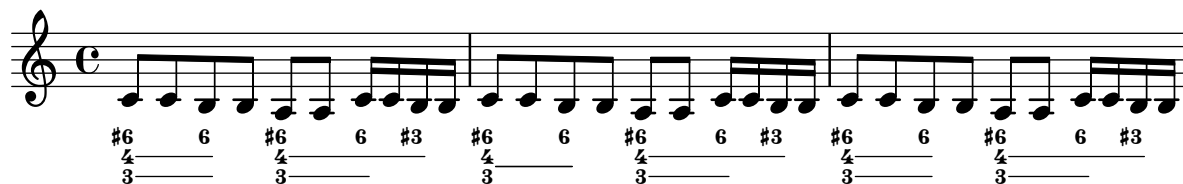
Where figured bass extender lines are being used by setting `useBassFigureExtenders` to `#t`, pairs of congruent figured bass extender lines are vertically centered if `figuredBassCenterContinuations` is set to `#t`.

```

<<
\relative c' {
  \repeat unfold 3 {
    c8 c b b a a c16 c b b
  }
}
\figures {
  \set useBassFigureExtenders = ##t
  <6+ 4 3>4 <6 4 3>8 r
  <6+ 4 3>4 <6 4 3>8 <4 3+>16 r
  \set figuredBassCenterContinuations = ##t
  <6+ 4 3>4 <6 4 3>8 r
  <6+ 4 3>4 <6 4 3>8 <4 3+>16 r
  \set figuredBassCenterContinuations = ##f
  <6+ 4 3>4 <6 4 3>8 r
}

```

<6+ 4 3>4 <6 4 3>8 <4 3+>16 r  
}  
>>

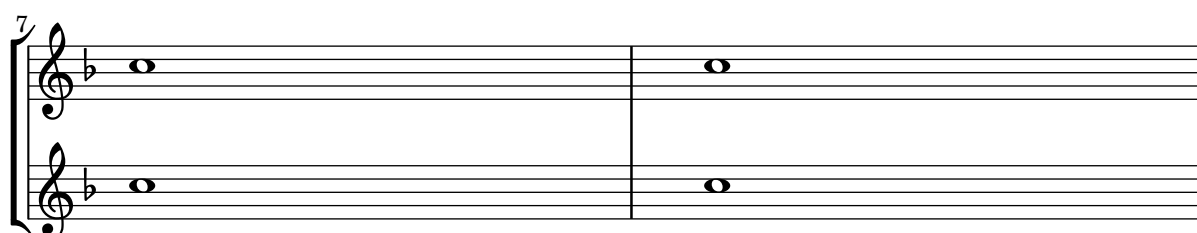
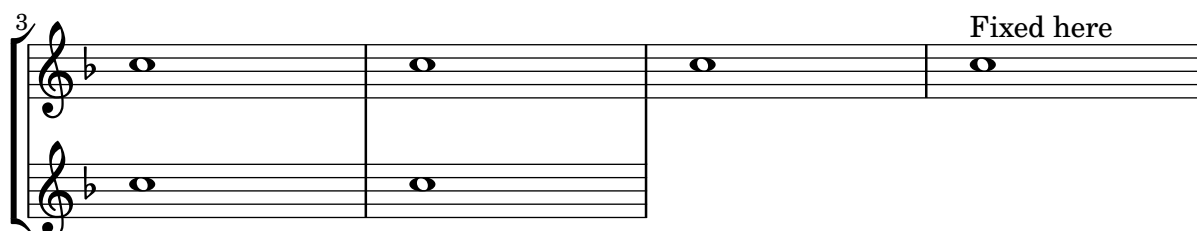


## 39 Workaround

### Adding an extra staff at a line break

When adding a new staff at a line break, some extra space is unfortunately added at the end of the line before the break (to fit in a key signature change, which will never be printed anyway). The workaround is to set the `explicitKeySignatureVisibility` property of the `Staff` grob as is shown in the example.

```
\score {
  \new StaffGroup \relative c'' {
    \new Staff
    \key f \major
    c1 c^"Unwanted extra space" \break
    << { c1 | c }
    \new Staff {
      \key f \major
      \once \omit Staff.TimeSignature
      c1 | c
    }
  }
  >>
  c1 | c^"Fixed here" \break
  << { c1 | c }
  \new Staff {
    \once \set Staff.explicitKeySignatureVisibility =
      #end-of-line-invisible
    \key f \major
    \once \omit Staff.TimeSignature
    c1 | c
  }
  >>
}
}
```





## Appoggiatura or grace note before a bar line

By default, appoggiaturas and grace notes that occur on the first beat of a measure are printed after the bar line. A possible solution for single staves to print it before the bar line is to add an invisible bar line and then the visible one.

In multi-staff systems, however, adding an invisible bar line distorts the positioning of full-bar rests in other staves; they are no longer centered but slightly shifted to the left. A better solution for such situations is to use the `\afterGrace` command with setting `afterGraceFraction` appropriately.

```
<<
{
  \appoggiatura d''8 c''4 r2. |
  \appoggiatura { \bar "" d''8 \bar "" } |
  c''4 r2.
}
{ R1 | R1 }
>>
```

`afterGraceFraction = 15/16`

```
<<
{
  \appoggiatura d''8 c''4 \afterGrace r2. d''8( |
  c''4) r2.
}
{ R1 | R1 }
>>
```

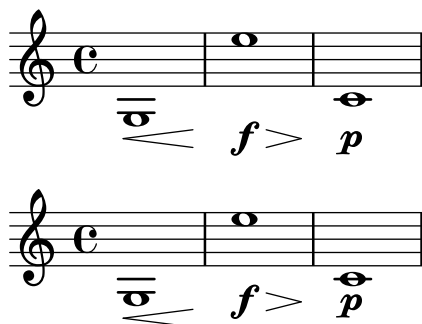


## Breaking vertical alignment of dynamics and textscripts

By default, LilyPond uses `DynamicLineSpanner` grobs to vertically align successive dynamic objects like hairpins and dynamic text. However, this is not always wanted. By inserting `\breakDynamicSpan`, which ends the alignment spanner prematurely, this vertical alignment can be avoided.

```
{ g1\< |
  e''\f\> |
  c'\p }
```

```
{ g1\< |
  e''\breakDynamicSpan\f\> |
  c'\p }
```



## Changing time signatures inside a polymetric section using `\scaleDurations`

### *Flexible polymeter with unaligned measures*

To support explicit creation of independently measured contexts, remove the `Timing_translator` from `Score` context and define a `TimingStaffGroup` context that has `Timing_translator`. This makes `Timing` an alias for `TimingStaffGroup`, targeting `\time` commands to the enclosing `TimingStaffGroup`.

Unlike LilyPond's built-in `\enablePerStaffTiming` command, this approach requires the explicit creation of `TimingStaffGroup` contexts; in exchange, it allows creating multiple `Staff` contexts that jointly follow the measure defined in their enclosing `TimingStaffGroup`.

### *Locally scaled time signatures*

Use the unscalable `\time` command to establish a measure of the desired length in `Timing`, a.k.a. `TimingStaffGroup`. In this snippet, all staves below `TimingStaffGroup` use a scaled time signature, so any time signature with the desired measure length is as good as any other. If there were an enclosed context that did not use a scaled time signature, the choice of time signature to set in `Timing` would matter in that context.

Use the `\polymetric \time` command to set scalable metric properties in contexts below `Timing`, and use the `\scaleDurations` command to scale both the local meter and the notes to fit the measure.

```
\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \accepts TimingStaffGroup
  }
  \context {
    \StaffGroup
    \name TimingStaffGroup
    \alias StaffGroup
    \consists "Timing_translator"
  }
}
```

<<

```

\new TimingStaffGroup <<
  \new Staff {
    \scaleDurations 8/5 {
      \time 6/5 % to set measure length in Timing
      \context Staff \polymetric \time 6/8
      b8 b b b b b b
      \time 4/5 % to set measure length in Timing
      \context Staff \polymetric \time 2/4
      b4 b
    }
  }
>>
\new TimingStaffGroup <<
  \new Staff {
    \clef bass
    \time 2/4
    c2 d e f
  }
>>
>>

```



## Creating “real” parenthesized dynamics

Although the easiest way to add parentheses to a dynamic mark is to use a `\markup` block, this method has a downside: the created objects behave like text markups and not like dynamics.

However, it is possible to create a similar object using the equivalent Scheme code (as described in the Notation Reference), combined with the `make-dynamic-script` function. This way, the markup is regarded as a dynamic and therefore remains compatible with commands such as `\dynamicUp` or `\dynamicDown`.

```

paren =
#(define-event-function (dyn) (ly:event?)
  (make-dynamic-script
    #{ \markup \concat {
      \normal-text \italic \fontsize #2 (
        \pad-x #0.2 #(ly:music-property dyn 'text)
      \normal-text \italic \fontsize #2 )
    }
    #}))

\relative c' {
  c4\paren\f c c \dynamicUp c\paren\p
}

```



## Cross-staff chords – beaming problems workaround

Sometimes it is better to use stems from the ‘other’ staff for creating cross-staff chords to trick LilyPond’s beam collision detector. In the following snippet, if the stems from the lower staff were used instead, it would be necessary to explicitly use

```
\override Staff.Beam.collision-voice-only = ##t
```

so that LilyPond doesn’t move the beams.

```
\new PianoStaff <<
  \new Staff = up \relative c' <<
    { r4
      \override Stem.cross-staff = ##t
      \override Stem.length = #19 % this is in half-spaces,
        % so it makes stems 9.5 staffspaces long
      \override Stem.Y-offset = #-6 % stems are normally lengthened
        % upwards, so here we must lower the stem by the amount
        % equal to the lengthening - in this case (19 - 7) / 2
        % (7 is default stem length)
      e e e }
    { s4
      \change Staff = "bottom"
      \override NoteColumn.ignore-collision = ##t
      c, c c
    }
  >>

  \new Staff = bottom \relative c' {
    \clef bass
    \voiceOne
    g8 a g a g a g a
  }
  >>
```



## Displaying complex chords

Here is a way to display a chord where the same note is played twice with different accidentals.

```
fixA = {
  \once \override Stem.length = #12
}
```

```
fixB = {
```

```

\once \override NoteHead.X-offset = #1.7
\once \override Stem.length = #7
\once \override Stem.rotation = #'(45 0 0)
\once \override Stem.extra-offset = #'(-0.1 . -0.2)
\once \override Flag.style = #'no-flag
\once \override Accidental.extra-offset = #'(4 . -.1)
}

\relative c' {
  << { \fixA <b d!>8 } \ { \voiceThree \fixB dis } >> s
}

```



## Extending glissandi across repeats

A glissando that 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 that takes the pitch of the grace note as its argument.

Note that in polyphonic music the grace note must be matched with corresponding grace notes in all other voices.

```

repeatGliss = #(define-music-function (grace)
  (ly:pitch?)
  #{
    % the next two lines ensure the glissando is long enough
    % to be visible
    \once \override Glissando.springs-and-rods
      = #ly:spanner::set-spacing-rods
    \once \override Glissando.minimum-length = 3.5
    \once \hideNotes
    \grace $grace \glissando
  #})

\score {
  \relative c' {
    \repeat volta 3 { c4 d e f\glissando }
    \alternative {
      { g2 d }
      { \repeatGliss f g2 e }
      { \repeatGliss f e2 d }
    }
  }
}

music = \relative c' {
  \voiceOne
  \repeat volta 2 {
    g a b c\glissando
  }
}

```

```

}
\alternative {
  { d1 }
  { \repeatGliss c \once \omit StringNumber e1\2 }
}
}

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

```

The image displays a musical score for a guitar. The top staff is a treble clef staff with a melody. The bottom staff is a guitar tablature staff. The melody consists of three measures, each with a first ending bracket. The tablature consists of two measures, each with a first ending bracket. The tablature includes fingerings (0, 2, 0, 1, 3, 5) and a repeat sign.

## Forcing measure width to adapt to a metronome mark's width

By default, metronome marks do not influence horizontal spacing. This can be solved through a simple override, as shown in the second half of the example.

```

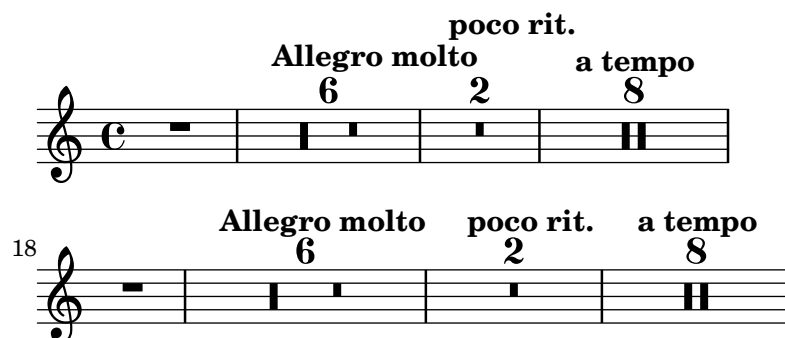
example = {
  R1
  \tempo "Allegro molto" R1*6
  \tempo "poco rit." R1*2
  \tempo "a tempo" R1*8 \break
}

{
  \compressMMRests {
    \example
    \override Score.MetronomeMark.extra-spacing-width = #'(-3 . 0)
    \example
  }
}

\layout {
  ragged-right = ##t
}

```

}



## 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'
}
```



## Marking notes of spoken parts with a cross on the stem (Sprechstimme)

This example shows how to put crosses on stems. Mark the beginning of a spoken section with the command `\speakOn` and end it with `\speakOff`.

```
speakOn = \override Stem.stencil =
  #(lambda (grob)
    (let* ((x-parent (ly:grob-parent grob X))
      (is-rest? (ly:grob? (ly:grob-object x-parent 'rest))))
      (if is-rest?
        empty-stencil
        (ly:stencil-combine-at-edge
          (ly:stem::print grob)
          Y
          (- (ly:grob-property grob 'direction))
          (grob-interpret-markup
            grob
            (markup #:center-align #:fontsize -4
              #:musicglyph "noteheads.s2cross")))
          -1.7))))
```

```
speakOff = \revert Stem.stencil
```

```
\new Staff {
  \relative c' {
```

```

a4 b a c
\peakOn
g4 f r g8 a
b4 r r8 d e4
\peakOff
c4 a g f
}
}

```



## 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.

Due to issue #6876 (<https://gitlab.com/lilypond/lilypond/-/issues/6876>), however, accidentals are ignored if this property is set. This snippet gives a workaround to circumvent the problem.

Another unfortunate side effect of this property is that LilyPond does not check whether there is enough horizontal space for grace notes (this is tracked as issue #2630 (<https://gitlab.com/lilypond/lilypond/-/issues/2630>)). You have to make sure that enough space is available, for example, by using `\newSpacingSection` together with a proper value for the `base-shortest-duration` of the `SpacingSpanner` grob.

```

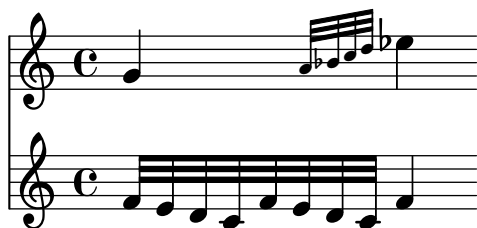
shiftedGrace =
#(define-music-function (offset music) (number? ly:music?)
  #{
    \override NoteHead.X-offset = #(- offset 0.85)
    \override Stem.X-offset = #offset
    \grace { $music }
    \revert NoteHead.X-offset
    \revert Stem.X-offset
  #})

\relative c' ' <<
  { g4 \shiftedGrace #-1.3 a32 \shiftedGrace #-0.5 { bes c d } es4 }
  { f,32 e d c f e d c f4 }
>>

\layout {
  \context {
    \Score
    \override SpacingSpanner.strict-grace-spacing = ##t
  }
}

```





## Positioning segno and coda (with line break)

If you want to place an exiting segno sign and add text like “D.S. al Coda” next to it where usually the staff lines are you can use this snippet. The coda will resume in a new line. There is a variation documented in this snippet, where the coda will remain on the same line.

```
\relative c' ' {
  c4 c c c | c c c c |
  \repeat segno 2 {
    c4 c c c | c c c c |
    \alternative {
      \volta 1 {
        c4 c c c | c c c c |
        % If you don't use \break at Coda, use \noBreak here
        % and after \bar "" below.
        \noBreak
        \section % double bar line
        \cadenzaOn % pause bar count
        \stopStaff % remove staff lines
        % Increasing the unfold counter will expand the staff-free space
        \repeat unfold 4 {
          s1
          \bar ""
        }
        % Place JumpScript where the staff would normally be.
        \once \override Score.JumpScript.outside-staff-priority = ##f
        \once \override Score.JumpScript.Y-offset = 0
        \startStaff % resume bar count
        \cadenzaOff % show staff lines again
      }
    }
  }
}
\sectionLabel "Coda"
% Show Coda on a new line
\break
\repeat unfold 6 { c4 c c c }
\fine
}
```





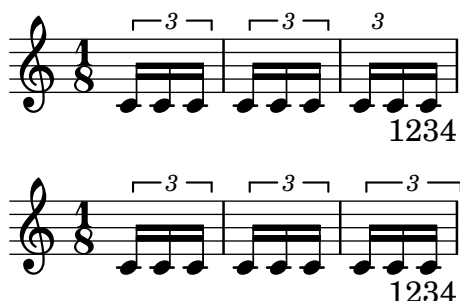
## Preventing final mark from removing final tuplet

Due to issue #2362 (<https://gitlab.com/lilypond/lilypond/-/issues/2362>) the addition of a final mark can result in the loss of a final tuplet marking. This can be overcome by setting `TupletBracket.full-length-to-extent` to `#f`.

```
\new Staff {
  \set tupletFullLength = ##t
  \time 1/8
  \tuplet 3/2 8 { c'16 c' c' c' c' c' c' c' c' }
  \tweak direction #DOWN \textEndMark "1234"
}

\new Staff {
  \set tupletFullLength = ##t
  \override TupletBracket.full-length-to-extent = ##f

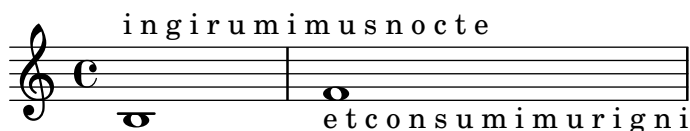
  \time 1/8
  \tuplet 3/2 8 { c'16 c' c' c' c' c' c' c' c' }
  \tweak direction #DOWN \textEndMark "1234"
}
```



## Printing text from right to left

It is possible to print text from right to left in a markup object, as demonstrated here.

```
{
  b1^\markup {
    \line { i n g i r u m i m u s n o c t e }
  }
  f'_\markup {
    \override #'(text-direction . -1)
    \line { i n g i r u m i m u s n o c t e }
  }
}
```



## 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  $\rightarrow$  c
- e sharp  $\rightarrow$  f
- c flat  $\rightarrow$  b
- f flat  $\rightarrow$  e

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

```
#(define (naturalize-pitch p)
  (let ((o (ly:pitch-octave p))
        ;; `ly:pitch-alteration` returns quarter tone steps.
        (a (* 4 (ly:pitch-alteration p)))
        (n (ly:pitch-notename p)))
    (cond
      ((and (> a 1)
            (or (eqv? n 6) (eqv? n 2))))
      (set! a (- a 2))
      (set! n (+ n 1)))
      ((and (< a -1)
            (or (eqv? n 0) (eqv? 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))))
    (when (< n 0)
      (set! o (- o 1))
      (set! n (+ n 7)))
    (when (> n 6)
      (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)))
    (when (pair? es)
      (ly:music-set-property! music 'elements
                              (map naturalize es)))
    (when (ly:music? e)
      (ly:music-set-property! music 'element
                              (naturalize e)))
    (when (ly:pitch? p)
      (set! p (naturalize-pitch p))
```

```

      (ly:music-set-property! music 'pitch p))
    music))

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

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

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

```



## Unfolding tremolo repeats

Currently, `note:duration`, which is a more or less a shortcut for `\repeat tremolo`, is not unfolded by `\unfoldRepeats` (this is tracked in issue #6145 (<https://gitlab.com/lilypond/lilypond/-/issues/6145>)). The function given in this snippet provides a workaround.

```

fixTremolos =
#(define-music-function (music) (ly:music?)
  (music-map
    (lambda (m)
      (let ((event (any (lambda (a)
                          (and (music-is-of-type? a 'tremolo-event)
                              a))
                        (ly:music-property m 'articulations)))))
        (if event
          (let* ((total-tremolo-duration (ly:music-property m
                                                             'duration))
                 (tremolo-type (ly:music-property event
                                                    'tremolo-type))
                 (one-tremolo-note-duration
                  (ly:make-duration (ly:intlog2 tremolo-type)))
                 (tremolo-note-count
                  (/ tremolo-type (expt 2 (ly:duration-log
                                           total-tremolo-duration)))))
            (set! (ly:music-property m 'duration)
                  one-tremolo-note-duration)
            (set! (ly:music-property m 'articulations)
                  (delete! event (ly:music-property m 'articulations)))
            (make-music 'TremoloRepeatedMusic
                        'repeat-count tremolo-note-count
                        'element m))
          m)))
    music))

```

```

unfoldRepeats = \unfoldRepeats #'() \fixTremolos \etc

music = { \repeat tremolo 8 c'16 c'2:16 }

{
  \music
  \unfoldRepeats \music
}

```



## Using an extra voice for breaks

Often it is easier to manage line and page-breaking information by keeping it separate from the music by introducing an extra voice containing only skips along with the `\break`, `\pageBreak`, and other layout information.

This pattern becomes especially helpful when overriding `line-break-system-details` and the other useful but long properties of the `NonMusicalPaperColumn` grob.

```
music = \relative c'' { c4 c c c }
```

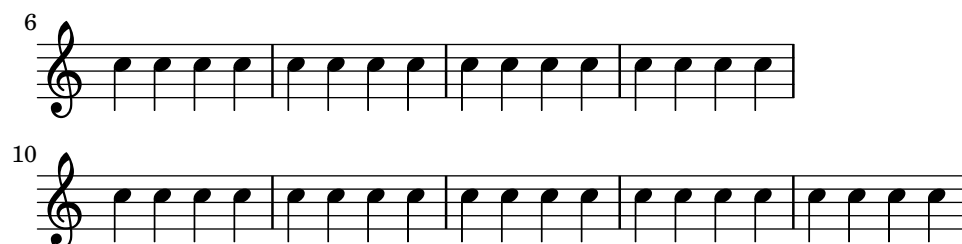
```

\score {
  \new Staff <<
    \new Voice {
      s1*2 \break
      s1*3 \break
      s1*4 \break
      s1*5 \break
    }
    \new Voice {
      \repeat unfold 2 { \music }
      \repeat unfold 3 { \music }
      \repeat unfold 4 { \music }
      \repeat unfold 5 { \music }
    }
  >>
}

\paper {
  indent = 0
  line-width = 140\mm
  ragged-right = ##t
}

```





## Vertically aligned dynamics and textscripts

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

The same idea, together with `\textLengthOn`, is used to align the text scripts along their baseline.

```
music = \relative c' {
  a'2\p b\f
  e4\p f\f\> g, b\p
  c2^\markup { \huge gorgeous } c^\markup { \huge fantastic }
}
```

```
{
  \music
  \break
  \override DynamicLineSpanner.staff-padding = 3
  \textLengthOn
  \override TextScript.staff-padding = 1
  \music
}
```

```
\paper { tagline = ##f }
```

