

LilyPond

The music typesetter

Snippets

The LilyPond development team

This document shows a selected set of LilyPond snippets from the LilyPond Snippet Repository (<http://lsr.di.unimi.it>) (LSR). It is in the public domain.

We would like to address many thanks to Sebastiano Vigna for maintaining LSR web site and database, and the University of Milano for hosting LSR.

Please note that this document is not an exact subset of LSR: some snippets come from `input/new` LilyPond sources directory, and snippets from LSR are converted through `convert-ly`, as LSR is based on a stable LilyPond version, and this document is for version 2.22.0.

Snippets are grouped by tags; tags listed in the table of contents match a section of LilyPond notation manual. Snippets may have several tags, and not all LSR tags may appear in this document.

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.

For more information about how this manual fits with the other documentation, or to read this manual in other formats, see Section “Manuals” in *General Information*.

If you are missing any manuals, the complete documentation can be found at <http://lilypond.org/>.

This document has been placed in the public domain.

For LilyPond version 2.22.0

Table of Contents

Pitches	1
Adding ambitus per voice	1
Adding an ottava marking to a single voice	1
Aiken head thin variant noteheads	2
Altering the length of beamed stems	2
Ambitus after key signature	3
Ambitus with multiple voices	3
Ambitus	3
Applying note head styles depending on the step of the scale	4
Automatically changing the stem direction of the middle note based on the melody	5
Changing ottava text	6
Changing the ambitus gap	6
Changing the interval of lines on the stave	7
Clefs can be transposed by arbitrary amounts	8
Coloring notes depending on their pitch	8
Creating a sequence of notes on various pitches	9
Creating custom key signatures	9
Force a cancellation natural before accidentals	10
Forcing a clef symbol to be displayed	10
Generating random notes	11
Hiding accidentals on tied notes at the start of a new system	11
Keep change clefs full sized	11
Makam example	12
Modifying the Ottava spanner slope	12
Non-traditional key signatures	13
Numbers as easy note heads	14
Orchestra choir and piano template	15
Preventing extra naturals from being automatically added	18
Preventing natural signs from being printed when the key signature changes	18
Quoting another voice with transposition	19
Separating key cancellations from key signature changes	20
Transposing pitches with minimum accidentals ("Smart" transpose)	20
Turkish Makam example	22
Tweaking clef properties	22
Using autochange with more than one voice	24
Rhythms	26
Adding beams, slurs, ties etc. when using triplet and non-triplet rhythms	26
Adding drum parts	26
Adjusting grace note spacing	27
Aligning bar numbers	27
Alternative breve notes	28
Appoggiatura or grace note before a bar line	28
Automatic beam subdivisions	29
Automatically change durations	29
Beam endings in Score context	30
Beams across line breaks	31
Changing beam knee gap	31

Changing form of multi-measure rests	32
Changing the number of augmentation dots per note	32
Changing the tempo without a metronome mark	32
Changing the tuplet number	33
Changing time signatures inside a polymetric section using <code>\scaleDurations</code>	33
Chant or psalms notation	34
Compound time signatures	34
Conducting signs, measure grouping signs	35
Consistently left aligned bar numbers	36
Controlling tuplet bracket visibility	37
Creating metronome marks in markup mode	38
Engraving ties manually	38
Engraving tremolos with floating beams	38
Entering several tuplets using only one <code>\tuplet</code> command	39
Flat flags and beam nibs	40
Forcing rehearsal marks to start from a given letter or number	41
Generating custom flags	41
Guitar strum rhythms	42
Heavily customized polymetric time signatures	43
Making an object invisible with the <code>'transparent</code> property	44
Making slurs with complex dash structure	45
Manually controlling beam positions	45
Merging multi-measure rests in a polyphonic part	46
Modifying tuplet bracket length	46
Moving dotted notes in polyphony	47
Multi-measure rest length control	47
Multi-measure rest markup	48
Non-default tuplet numbers	48
Numbering single measure rests	49
Partcombine and <code>autoBeamOff</code>	50
Percussion example	50
Permitting line breaks within beamed tuplets	52
Positioning grace note beams at the height of normal note beams	52
Positioning grace notes with floating space	53
Positioning multi-measure rests	53
Preventing final mark from removing final tuplet	54
Printing bar numbers at regular intervals	55
Printing bar numbers inside boxes or circles	55
Printing bar numbers with changing regular intervals	56
Printing metronome and rehearsal marks below the staff	56
Printing music with different time signatures	57
Printing the bar number for the first measure	60
Redefining grace note global defaults	61
Removing bar numbers from a score	61
Removing connecting bar lines on <code>StaffGroup</code> , <code>PianoStaff</code> , or <code>GrandStaff</code>	62
Rest styles	62
Reverting default beam endings	63
Rhythmic slashes	64
Skips in lyric mode (2)	64
Skips in lyric mode	65
Stemlets	65
Strict beat beaming	66
Subdividing beams	66
Tam-tam example	67

Three-sided box.....	67
Time signature in parentheses - method 3	68
Time signature in parentheses.....	68
Time signature printing only the numerator as a number (instead of the fraction)	68
Tweaking grace layout within music	69
User defined time signatures	70
Using alternative flag styles	70
Using grace note slashes with normal heads	71
Using ties with arpeggios.....	71
Expressive marks	72
Adding beams, slurs, ties etc. when using tuplet and non-tuplet rhythms	72
Adding parentheses around an expressive mark or chordal note	72
Adding timing marks to long glissandi.....	72
Adjusting the shape of falls and doits	73
Alternative breve notes	74
Asymmetric slurs	74
Breathing signs	74
Broken Crescendo Hairpin	75
Caesura ("railtracks") with fermata	76
Center text below hairpin dynamics	76
Changing \flageolet mark size	78
Changing text and spanner styles for text dynamics	78
Changing the appearance of a slur from solid to dotted or dashed.....	79
Changing the breath mark symbol	79
Changing the number of augmentation dots per note	79
Combining dynamics with markup texts.....	80
Contemporary glissando.....	80
Controlling spanner visibility after a line break	80
Controlling the vertical ordering of scripts.....	81
Creating a delayed turn.....	81
Creating arpeggios across notes in different voices	82
Creating cross-staff arpeggios in a piano staff.....	82
Creating cross-staff arpeggios in other contexts	83
Creating double-digit fingerings	84
Creating "real" parenthesized dynamics	84
Creating simultaneous rehearsal marks	84
Creating slurs across voices	85
Creating text spanners.....	86
Dynamics custom text spanner postfix	87
Dynamics text spanner postfix	87
Glissandi can skip grobs	88
Hairpins with different line styles.....	88
Hiding the extender line for text dynamics	88
Horizontally aligning custom dynamics (e.g. "sempre pp" "piu f" "subito p")	89
How to print two rehearsal marks above and below the same barline (method 1)	92
How to print two rehearsal marks above and below the same barline (method 2)	93
Inserting a caesura.....	94
Laissez vibrer ties.....	94
Line arrows.....	95
Making slurs with complex dash structure.....	95
Modifying default values for articulation shorthand notation.....	96
Moving slur positions vertically	96
Moving the ends of hairpins.....	97

Positioning arpeggios	98
Positioning text markups inside slurs	98
Printing hairpins in various styles	98
Printing hairpins using al niente notation	99
Printing metronome and rehearsal marks below the staff	99
Setting hairpin behavior at bar lines	100
Setting the minimum length of hairpins	100
Showing the same articulation above and below a note or chord	100
Snap-pizzicato or Bartok pizzicato	101
Using a tick as the breath mark symbol	101
Using arpeggioBracket to make divisi more visible	102
Using double slurs for legato chords	102
Using the whiteout property	103
Vertical line as a baroque articulation mark	103
Vertically aligning dynamics across multiple notes	103
Repeats	105
Adding volta brackets to additional staves	105
Centered measure numbers	105
Changing the default bar lines	106
Cross-staff tremolos	107
Engraving tremolos with floating beams	107
Isolated percent repeats	108
Measure counter	108
Numbering groups of measures	109
Percent repeat count visibility	110
Percent repeat counter	110
Positioning segno and coda (with line break)	111
Setting the double repeat default for volte	113
Shortening volta brackets	113
Volta below chords	114
Volta multi staff	114
Volta text markup using repeatCommands	115
Simultaneous notes	117
Additional voices to avoid collisions	117
Changing a single note's size in a chord	117
Changing partcombine texts	118
Clusters	118
Combining two parts on the same staff	119
Displaying complex chords	120
Forcing horizontal shift of notes	120
Making an object invisible with the 'transparent property	121
Moving dotted notes in polyphony	121
Suppressing warnings for clashing note columns	122
Two \partCombine pairs on one staff	122

Staff notation	125
Adding ambitus per voice	125
Adding an extra staff at a line break	125
Adding an extra staff	126
Adding indicators to staves which get split after a break	127
Adding orchestral cues to a vocal score	131
Adding timing marks to long glissandi	133
Alternative bar numbering	133
Ambitus after key signature	134
Centered measure numbers	135
Changing the default bar lines	136
Changing the number of lines in a staff	136
Changing the staff size	137
Creating blank staves	137
Creating custom key signatures	139
Creating double-digit fingerings	140
Cross staff stems	140
Display bracket with only one staff in a system	141
Extending a TrillSpanner	141
Extending glissandi across repeats	142
Flat Ties	143
Forcing measure width to adapt to MetronomeMark's width	146
Glissandi can skip grobs	146
How to print two rehearsal marks above and below the same barline (method 1)	146
How to print two rehearsal marks above and below the same barline (method 2)	147
Incipit	148
Inserting score fragments above a staff, as markups	153
Let TabStaff print the topmost string at bottom	153
Letter tablature formatting	154
Making glissandi breakable	155
Making some staff lines thicker than the others	155
Measure counter	156
Mensurstriche layout (bar lines between the staves)	156
Modifying the Ottava spanner slope	157
Nesting staves	157
Non-traditional key signatures	158
Numbering groups of measures	159
Orchestra choir and piano template	160
Putting lyrics inside the staff	163
Quoting another voice with transposition	164
Quoting another voice	164
Removing brace on first line of piano score	165
Removing the first empty line	166
Setting system separators	167
Tick bar lines	170
Time signature in parentheses - method 3	170
Time signature in parentheses	170
Tweaking clef properties	170
Two \partCombine pairs on one staff	172
Use square bracket at the start of a staff group	174
Using autochange with more than one voice	174
Using marklines in a Frenched score	175
Vertical aligned StaffGroups without connecting SystemStartBar	178
Volta below chords	186

Volta multi staff	187
Editorial annotations	188
Adding fingerings to a score	188
Adding links to objects	188
Adding markups in a tablature	189
Allowing fingerings to be printed inside the staff	190
Alternative bar numbering	191
Analysis brackets above the staff	192
Analysis brackets with labels	192
Applying note head styles depending on the step of the scale	193
Blanking staff lines using the \whiteout command	194
Changing a single note's size in a chord	194
Changing the appearance of a slur from solid to dotted or dashed	194
Coloring notes depending on their pitch	195
Controlling the placement of chord fingerings	196
Creating a delayed turn	196
Creating blank staves	197
Creating double-digit fingerings	199
Default direction of stems on the center line of the staff	199
Different font size settings for instrumentName and shortInstrumentName	199
Drawing boxes around grobs	201
Drawing circles around note heads	201
Drawing circles around various objects	202
Embedding native PostScript in a \markup block	202
Grid lines: changing their appearance	202
Grid lines: emphasizing rhythms and notes synchronization	203
Hammer on and pull off using chords	205
Hammer on and pull off using voices	205
Hammer on and pull off	205
How to print two rehearsal marks above and below the same barline (method 1)	206
How to print two rehearsal marks above and below the same barline (method 2)	206
Making some staff lines thicker than the others	207
Marking notes of spoken parts with a cross on the stem	208
Measure counter	208
Numbering groups of measures	209
Positioning fingering indications precisely	210
Positioning text markups inside slurs	211
Printing text from right to left	211
String number extender lines	211
Using PostScript to generate special note head shapes	212
Using the whiteout property	212
Text	214
Adding markups in a tablature	214
Adding the current date to a score	214
Adjusting lyrics vertical spacing	215
Aligning and centering instrument names	216
Aligning objects created with the \mark command	217
Aligning syllables with melisma	218
Blanking staff lines using the \whiteout command	218
Center text below hairpin dynamics	219
Changing ottava text	220

Changing the default text font family	221
Combining dynamics with markup texts	221
Combining two parts on the same staff	222
Creating "real" parenthesized dynamics	223
Creating simultaneous rehearsal marks	224
Creating text spanners	224
Demonstrating all headers	225
Embedding native PostScript in a \markup block	226
Formatting lyrics syllables	227
How to put ties between syllables in lyrics	227
Lyrics alignment	227
Markup list	228
Multi-measure rest markup	230
Of the ubiquity of markup objects	230
Outputting the version number	231
Piano template with centered lyrics	232
Printing bar numbers with changing regular intervals	232
Printing marks at the end of a line	233
Printing marks on every staff	233
Printing text from right to left	234
Putting lyrics inside the staff	234
Stand-alone two-column markup	235
String number extender lines	235
Three-sided box	236
UTF-8	237
Vocal ensemble template with lyrics aligned below and above the staves	238
Volta text markup using repeatCommands	240
Vocal music	241
Adding ambitus per voice	241
Adding indicators to staves which get split after a break	241
Adding orchestral cues to a vocal score	245
Adjusting lyrics vertical spacing	247
Aligning syllables with melisma	247
Ambitus after key signature	248
Ambitus with multiple voices	248
Ambitus	249
Ancient notation template – modern transcription of gregorian music	250
Anglican psalm template	250
Arranging separate lyrics on a single line	253
Changing stanza fonts	254
Chant or psalms notation	255
Forcing hyphens to be shown	255
Formatting lyrics syllables	256
How to put ties between syllables in lyrics	256
Hymn template	257
Lyrics alignment	259
Marking notes of spoken parts with a cross on the stem	259
Obtaining 2.12 lyrics spacing in newer versions	260
Orchestra choir and piano template	262
Piano template with melody and lyrics	266
Putting lyrics inside the staff	267
SATB Choir template - four staves	268
Single staff template with notes, lyrics, and chords	269

Single staff template with notes, lyrics, chords and frets	270
Single staff template with notes and lyrics	271
Skips in lyric mode (2)	272
Skips in lyric mode	272
Using arpeggioBracket to make divisi more visible	272
Using tags to produce mensural and modern music from the same source	273
Vertically aligning ossias and lyrics	275
Vertically centered common lyrics	275
Vocal ensemble template with automatic piano reduction	277
Vocal ensemble template with lyrics aligned below and above the staves	279
Vocal ensemble template with verse and refrain	280
Vocal ensemble template	282
Chords	285
Adding a figured bass above or below the notes	285
Adding bar lines to ChordNames context	285
Bar chords notation for Guitar (with Text Spanner)	286
Changing chord separator	287
Changing the chord names to German or semi-German notation	287
Changing the positions of figured bass alterations	288
Chord name exceptions	288
chord name major7	289
Chord names alternative	289
Chords with stretched fingering for FretBoards and TabVoice	299
Clusters	300
Controlling the placement of chord fingerings	300
Cross-staff chords - beaming problems workaround	301
Displaying complex chords	301
Manually break figured bass extenders for only some numbers	302
Showing chords at changes	302
Simple lead sheet	303
Single staff template with notes, lyrics, and chords	303
Single staff template with notes, lyrics, chords and frets	304
Single staff template with notes and chords	305
Vertically centering paired figured bass extenders	306
Volta below chords	306
Keyboards	308
Accordion-discant symbols	308
Accordion register symbols	311
Changing the text for sustain markings	312
Clusters	313
Controlling the placement of chord fingerings	313
Creating slurs across voices	314
Cross-staff chords - beaming problems workaround	314
Cross-staff tremolos	315
Fine-tuning pedal brackets	316
Indicating cross-staff chords with arpeggio bracket	316
Jazz combo template	317
Laissez vibrer ties	323
Piano template (simple)	324
Piano template with centered lyrics	324
Piano template with melody and lyrics	325

Removing brace on first line of piano score	326
Using autochange with more than one voice	327
Vocal ensemble template with automatic piano reduction	328
Percussion	331
Adding drum parts	331
Customized drum notation in printed and MIDI output	332
Heavily customized polymetric time signatures	334
Jazz combo template	335
Percussion beaters	341
Percussion example	344
Printing music with different time signatures	345
Tam-tam example	349
Fretted strings	350
Adding fingerings to a score	350
Adding fingerings to tablatures	350
Adding markups in a tablature	351
Allowing fingerings to be printed inside the staff	351
Barres in automatic fretboards	352
Bar chords notation for Guitar (with Text Spanner)	352
Changing fret orientations	353
Chord glissando in tablature	354
ChordChanges for FretBoards	355
Chords with stretched fingering for FretBoards and TabVoice	356
Controlling the placement of chord fingerings	356
Customizing fretboard fret diagrams	357
Customizing markup fret diagrams	358
Defining predefined fretboards for other instruments	360
Faking a hammer in tablatures	362
Fingerings, string indications, and right-hand fingerings	362
Flamenco notation	362
Fret diagrams explained and developed	366
Fretboards alternate tables	373
Fretted-string harmonics in tablature	374
Guitar slides	376
Guitar strum rhythms	377
Hammer on and pull off using chords	378
Hammer on and pull off using voices	378
Hammer on and pull off	379
How to change fret diagram position	379
Jazz combo template	380
Laissez vibrer ties	386
Let TabStaff print the topmost string at bottom	387
Letter tablature formatting	387
Open string harmonics in tablature	388
Placement of right-hand fingerings	390
Polyphony in tablature	391
Slides in tablature	391
Stem and beam behavior in tablature	392
String number extender lines	393

Unfretted strings	394
Creating slurs across voices	394
Dotted harmonics	394
Snap-pizzicato or Bartok pizzicato	395
String quartet template (simple)	395
String quartet template with separate parts	396
Winds	400
Changing the size of woodwind diagrams	400
Fingering symbols for wind instruments	400
Flute slap notation	401
Graphical and text woodwind diagrams	402
Recorder fingering chart	402
Woodwind diagrams key lists	403
Woodwind diagrams listing	404
Ancient notation	407
Adding a figured bass above or below the notes	407
Ancient fonts	407
Ancient notation template – modern transcription of gregorian music	412
Ancient time signatures	413
Chant or psalms notation	413
Custodes	414
Incipit	415
Mensurstriche layout (bar lines between the staves)	419
Rest styles	420
Using tags to produce mensural and modern music from the same source	421
Vertical line as a baroque articulation mark	422
World music	424
Arabic improvisation	424
Makam example	424
Non-traditional key signatures	424
Printing text from right to left	425
Turkish Makam example	425
Contexts and engravers	427
Adding a figured bass above or below the notes	427
Adding an extra staff at a line break	427
Adding an extra staff	428
Automatically changing the stem direction of the middle note based on the melody	429
Centered measure numbers	430
Changing MIDI output to one channel per voice	430
Changing time signatures inside a polymetric section using \scale Durations	431
Chant or psalms notation	432
Creating blank staves	433
Creating custom key signatures	435
Cross staff stems	435
Defining an engraver in Scheme: ambitus engraver	436
Displaying a whole GrandStaff system if only one of its staves is alive	442
Engravers one-by-one	444
Mensurstriche layout (bar lines between the staves)	449

Nesting staves	450
Numbering groups of measures	451
Removing bar numbers from a score	452
Use square bracket at the start of a staff group	452
Using marklines in a Frenched score	453
Vocal ensemble template with lyrics aligned below and above the staves	455
Vocal ensemble template with verse and refrain	457
Tweaks and overrides	460
Adding an ottava marking to a single voice	460
Adding links to objects	460
Adding markups in a tablature	462
Adding timing marks to long glissandi	462
Adjusting grace note spacing	463
Adjusting lyrics vertical spacing	464
Altering the length of beamed stems	464
Alternative bar numbering	465
Analysis brackets above the staff	466
Analysis brackets with labels	466
Asymmetric slurs	467
Caesura ("railtracks") with fermata	467
Changing a single note's size in a chord	468
Changing beam thickness and spacing	468
Changing form of multi-measure rests	469
Changing properties for individual grobs	469
Changing text and spanner styles for text dynamics	469
Changing the default text font family	470
Changing the staff size	471
Changing the tempo without a metronome mark	471
Changing the text for sustain markings	472
Controlling spanner visibility after a line break	472
Controlling the vertical ordering of scripts	473
Controlling tuplet bracket visibility	473
Creating a delayed turn	474
Creating custom key signatures	474
Creating double-digit fingerings	475
Creating simultaneous rehearsal marks	475
Creating text spanners	476
Cross-staff chords - beaming problems workaround	477
Cross staff stems	478
Custodes	478
Customizing fretboard fret diagrams	479
Customizing markup fret diagrams	481
Display bracket with only one staff in a system	482
Displaying grob ancestry	483
Dotted harmonics	485
Drawing boxes around grobs	485
Drawing circles around various objects	486
Dynamics custom text spanner postfix	486
Dynamics text spanner postfix	487
Extending a TrillSpanner	487
Extending glissandi across repeats	488
Fine-tuning pedal brackets	489
Flat Ties	489

Force a cancellation natural before accidentals.....	492
Forcing horizontal shift of notes.....	492
Fret diagrams explained and developed.....	492
Generating custom flags.....	499
Glissandi can skip grobs.....	500
Hairpins with different line styles.....	501
Horizontally aligning custom dynamics (e.g. "sempre pp" "piu f" "subito p").....	501
How to change fret diagram position.....	505
How to print two rehearsal marks above and below the same barline (method 1).....	506
How to print two rehearsal marks above and below the same barline (method 2).....	507
Inserting a caesura.....	508
Keep change clefs full sized.....	508
Line arrows.....	509
Making an object invisible with the 'transparent property.....	509
Making glissandi breakable.....	510
Manually controlling beam positions.....	510
Mensurstriche layout (bar lines between the staves).....	511
Modifying the Ottava spanner slope.....	511
Moving dotted notes in polyphony.....	512
Moving slur positions vertically.....	512
Nesting staves.....	513
Overriding articulations of distinct type.....	514
Percent repeat count visibility.....	515
Positioning arpeggios.....	516
Positioning multi-measure rests.....	516
Positioning text markups inside slurs.....	517
Printing bar numbers inside boxes or circles.....	518
Printing metronome and rehearsal marks below the staff.....	518
Printing note names with and without an octave marker.....	519
Proportional strict notespacing.....	519
Removing brace on first line of piano score.....	520
Removing connecting bar lines on StaffGroup, PianoStaff, or GrandStaff.....	521
Removing the first empty line.....	521
Rest styles.....	522
Rhythmic slashes.....	524
Separating key cancellations from key signature changes.....	524
Setting hairpin behavior at bar lines.....	525
Setting system separators.....	525
Showing the same articulation above and below a note or chord.....	528
String number extender lines.....	528
Suppressing warnings for clashing note columns.....	529
Time signature in parentheses - method 3.....	529
Time signature in parentheses.....	530
Time signature printing only the numerator as a number (instead of the fraction).....	530
Tuplet bracket and change staff.....	530
Tweaking clef properties.....	531
Tweaking grace layout within music.....	533
Using alternative flag styles.....	533
Using ly:grob-object to access grobs with \tweak.....	534
Using PostScript to generate special note head shapes.....	535
Using the \tweak command to tweak individual grobs.....	536
Vertically aligned dynamics and textscripts.....	536
Vertically aligning ossias and lyrics.....	537
Vertically centering paired figured bass extenders.....	538

Paper and layout	539
Aligning and centering instrument names	539
Arranging separate lyrics on a single line	540
Book parts	541
Changing the staff size	545
Clip systems	546
Creating blank staves	549
Demonstrating all headers	550
Setting system separators	551
Table of contents	554
Vertical aligned StaffGroups without connecting SystemStartBar	555
Titles	564
Adding the current date to a score	564
Aligning and centering instrument names	564
Demonstrating all headers	566
Outputting the version number	567
Spacing	568
Adjusting lyrics vertical spacing	568
Allowing fingerings to be printed inside the staff	568
Page label	569
Proportional strict notespacing	571
Vertically aligned dynamics and textscripts	572
Vertically aligning ossias and lyrics	572
MIDI	574
Changing MIDI output to one channel per voice	574
Changing the tempo without a metronome mark	575
Creating custom dynamics in MIDI output	575
Customized drum notation in printed and MIDI output	576
Demo MidiInstruments	578
Replacing default MIDI instrument equalization	582
Templates	584
Ancient notation template – modern transcription of gregorian music	584
Anglican psalm template	585
Hymn template	587
Jazz combo template	589
Orchestra choir and piano template	595
Piano template (simple)	599
Piano template with centered lyrics	600
Piano template with melody and lyrics	601
SATB Choir template - four staves	602
Score for diatonic accordion	603
Single staff template with notes, lyrics, and chords	608
Single staff template with notes, lyrics, chords and frets	609
Single staff template with notes and chords	610
Single staff template with notes and lyrics	610
Single staff template with only notes	611
String quartet template (simple)	611
String quartet template with separate parts	613

Vocal ensemble template with automatic piano reduction	615
Vocal ensemble template with lyrics aligned below and above the staves	617
Vocal ensemble template with verse and refrain.....	619
Vocal ensemble template.....	621

Pitches

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

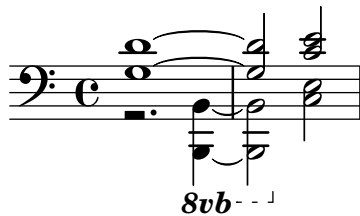
```
{
  \clef bass
  << { <g d'>1~ q2 <c' e'> }
  \\
  {
    r2.
    \set Staff.ottavation = #"8vb"
    \once \override Staff.OttavaBracket.direction = #DOWN
    \set Voice.middleCPosition = #(+ 6 7)
    <b,,, b,,,>4 ~ |
    q2
    \unset Staff.ottavation
    \unset Voice.middleCPosition
    <c e>2
```



```

    }
  >>
}

```



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
  }
}
% END EXAMPLE

```



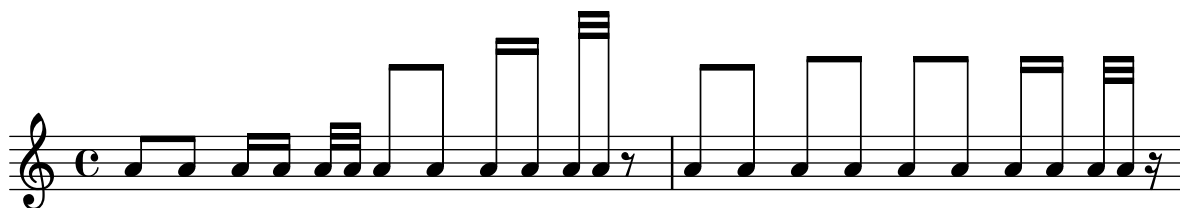
Altering the length of beamed stems

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
  a8[ a] a16[ a] a32[ a] r16
}

```



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 Section “Graphical Object Interfaces” in *Internals Reference* 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
  }
}>>
```



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'
  }
}
>>

```



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)

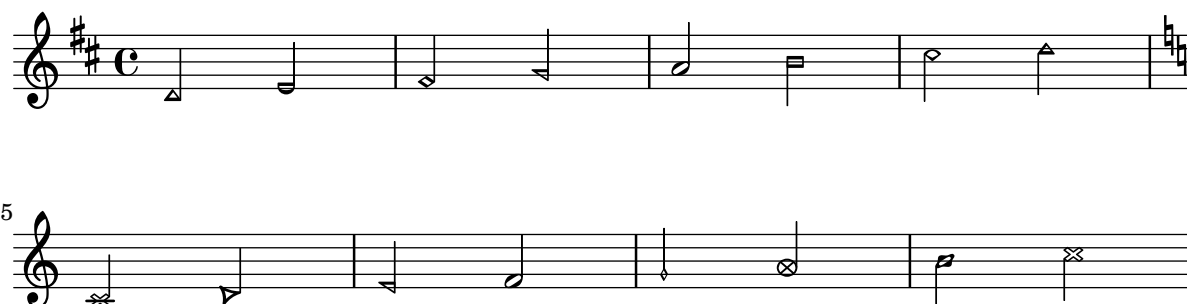
  \fragment
}

\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 and overriding the `neutral-direction` of `Stem`.

```

\relative c'' {
  \time 3/4
  a8 b g f b g |
  c b d c b c |
}

\layout {
  \context {
    \Voice
    \consists "Melody_engraver"
    \autoBeamOff
    \override Stem.neutral-direction = #'()
  }
}

```



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 with an extended range of printed flats.

```
\new Staff \with {
  \override StaffSymbol.line-count = #8
  \override KeySignature.flat-positions = #'((-7 . 6))
  \override KeyCancellation.flat-positions = #'((-7 . 6))
  % presumably sharps are also printed in both octaves
  \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,
}

```



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 24 random notes (or as many as required), based on the current time (or any randomish number specified instead, in order to obtain the same random notes each time): i.e., to get different random note patterns, just change this number.

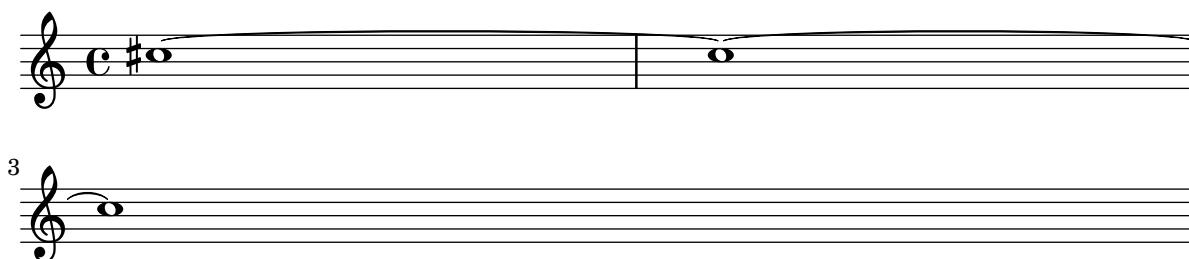
```
\score {
  {
    $(let ((random-state (seed->random-state (current-time))))
      (make-sequential-music
        (map (lambda (x)
              (let ((idx (random 12 random-state)))
                (make-event-chord
                  (list
                    (make-music 'NoteEvent
                              'duration (ly:make-duration 2 0 1/1)
                              'pitch (ly:make-pitch
                                     (quotient idx 7)
                                     (remainder idx 7)
                                     0)))))))
              (make-list 24))))))
  }
}
```



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 is changed, the clef sign displayed is smaller than the initial clef. This can be overridden with `full-size-change`.

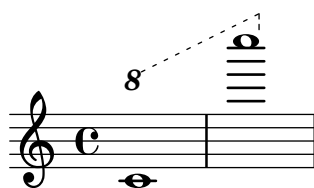
```
\relative c' {
  \clef "treble"
  c1
}
```



```

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

```



Non-traditional key signatures

The commonly used `\key` command sets the `keyAlterations` property in the `Staff` context. To create non-standard key signatures, set this property directly.

The format of this command is a list:

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

where, for each element in the list `octave` specifies the octave (0 being the octave from middle c to the b above), `step` specifies the note within the octave (0 means c and 6 means b), and `alter` is `,SHARP` `,FLAT` `,DOUBLE-SHARP` etc.

Alternatively, using the more concise format for each item in the list, `(step . alter)` specifies the same alteration holds in all octaves. For microtonal scales where a “sharp” is not 100 cents, `alter` refers to the 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 \dwn reb resd
  dod dob dosd \dwn dob |
  dobsb dods 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 26)

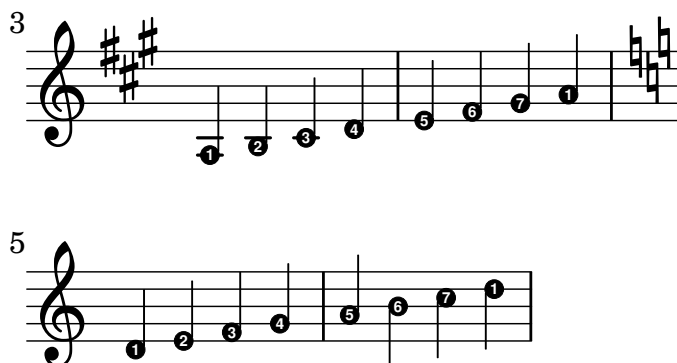
\layout {
  ragged-right = ##t
  \context {
    \Voice
    \consists \Ez_numbers_engraver
  }
}

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

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

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





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 }

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

altoILyrics = \sopranoLyrics

altoIIILyrics = \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 }
violinIMus = \relative c' { \key g \major g'1 b }
violinIIMus = \relative c' { \key g \major g'1 b }
violaMus = \relative c { \clef alto \key g \major g'1 b }
celloMus = \relative c { \clef bass \key g \major g1 b }
bassMus = \relative c { \clef "bass_8" \key g \major g,1 b }

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

      \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' \clarinetMus
    >>

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

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

    \new RhythmicStaff = "RhythmicStaff_percussion"
    \with { instrumentName = "Percussion" }
    <<
      \percussionMus
    >>

    \new PianoStaff \with { instrumentName = "Piano" }
    <<
      \new Staff { \pianoRHMus }
      \new Staff { \pianoLHMus }
  >>
}

```

```

>>
\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
>>
>>
\layout { }
}

```


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

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 to `f` the `printKeyCancellation` property in the `Staff` context.

```

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

```





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. This example only modifies the second list, moving key-cancellation before staff-bar; by modifying the second list, break alignment behavior only changes in the middle of a system, not at the beginning or the end.

```
\new Staff {
  \override Score.BreakAlignment.break-align-orders =
    ##((left-edge ambitus breathing-sign clef staff-bar
        key-cancellation key-signature time-signature custos)

        (left-edge ambitus breathing-sign clef key-cancellation
          staff-bar key-signature time-signature custos)

        (left-edge ambitus breathing-sign clef key-cancellation
          key-signature staff-bar time-signature custos))

  \key des \major
  c'1
  \bar "||"
  \key bes \major
  c'1
}
```



Transposing pitches with minimum accidentals ("Smart" transpose)

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

Double accidentals should be removed

B sharp -> C

E sharp -> F

C flat -> B

F flat -> E

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

```
#(define (naturalize-pitch p)
  (let ((o (ly:pitch-octave p))
        (a (* 4 (ly:pitch-alteration p))))
    ;; alteration, a, in quarter tone steps,
    ;; for historical reasons
    (n (ly:pitch-notename p))))
```

```

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

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

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

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

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

```



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.

```
% Initialize makam settings
\include "turkish-makam.ly"

\header {
  title = "Hüseyni Saz Semaisi"
  composer = "Lavtacı Andon"
}

\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 |
}
```

Hüseyni 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 `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

Using autochange with more than one voice.

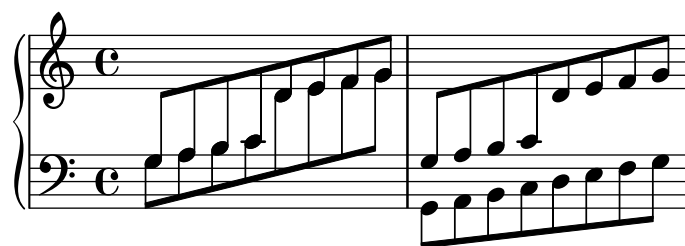
```

\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
}
>>
}

```



Rhythms

Section “Rhythms” in *Notation Reference*

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

LilyPond syntax can involve many unusual placements for parentheses, brackets etc., which might sometimes have to be interleaved.

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

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

```

\relative c' {

```

```

\set Score.currentBarNumber = #111
\override Score.BarNumber.break-visibility = #all-visible
% Increase the size of the bar number by 2
\override Score.BarNumber.font-size = #2
% Print a bar number every second measure
\set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
c1 | c1
% Center-align bar numbers
\override Score.BarNumber.self-alignment-X = #CENTER
c1 | c1
% Left-align bar numbers
\override Score.BarNumber.self-alignment-X = #LEFT
c1 | c1
}

```



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. They can however be printed before, simply by adding an invisible BarLine and then the visible one, as demonstrated here.

```

{
  R1
  %% default
  \appoggiatura d''8 c''4 r2.
  %% cheated
  \appoggiatura { \bar "" d''8 \bar "|" } c''4 r2.
}

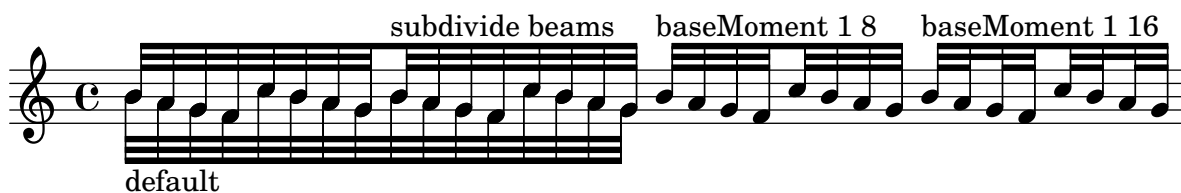
```



Automatic beam subdivisions

Beams can be subdivided automatically. By setting the property `subdivideBeams`, beams are subdivided at beat positions (as specified in `baseMoment`).

```
\new Staff {
  \relative c'' {
    <<
    {
      \voiceOne
      \set subdivideBeams = ##t
      b32[ a g f c' b a g
      b32^"subdivide beams" 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
    \set baseMoment = #(ly:make-moment 1/8)
    \set beatStructure = 2,2,2,2
    b32^"baseMoment 1 8"[ a g f c' b a g]
    \set baseMoment = #(ly:make-moment 1/16)
    \set beatStructure = 4,4,4,4
    b32^"baseMoment 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.

```
\paper { indent = 0 }
```

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

```
\score {
  \new Voice {
    \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:

```

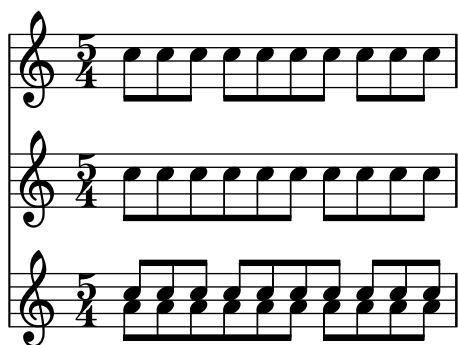
\relative c'' {
  \time 5/4
  % Set default beaming for all staves
  \set Score.baseMoment = #(ly:make-moment 1/8)
  \set Score.beatStructure = 3,4,3
  <<
    \new Staff {
      c8 c c c c c c c c c
    }
    \new Staff {
      % Modify beaming for just this staff
      \set Staff.beatStructure = 6,4
      c8 c c c c c c c c c 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
}
>>
}
>>
}

```



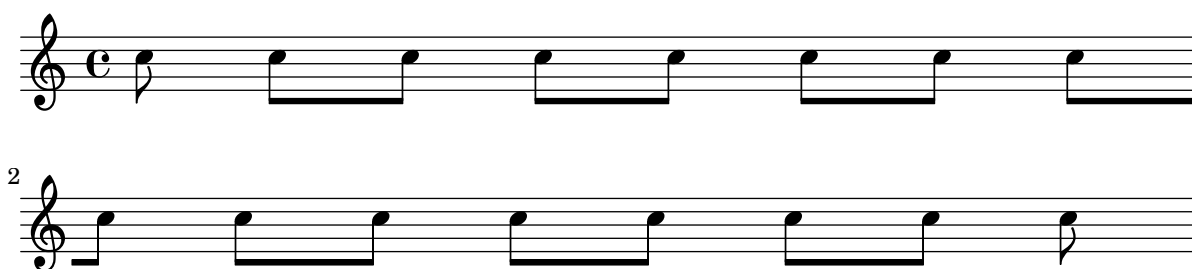
Beams across line breaks

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

```

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

```



Changing beam knee gap

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

```

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

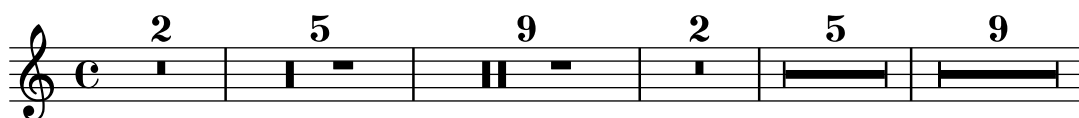
```



Changing form of multi-measure rests

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

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



Changing the number of augmentation dots per note

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

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



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 { }
}
```

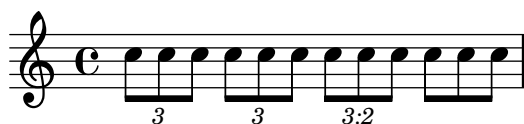


Changing the tuplet number

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`

The `measureLength` property, together with `measurePosition`, determines when a bar line is needed. However, when using `\scaleDurations`, the scaling of durations makes it difficult to change time signatures. In this case, `measureLength` should be set manually, using the `ly:make-moment` callback. The second argument must be the same as the second argument of `\scaleDurations`.

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

```
<<
  \new Staff {
```



```

\scaleDurations 8/5 {
  \time 6/8
  \set Timing.measureLength = #(ly:make-moment 6/5)
  b8 b b b b b
  \time 2/4
  \set Timing.measureLength = #(ly:make-moment 4/5)
  b4 b
}
}
\new Staff {
  \clef bass
  \time 2/4
  c2 d e f
}
>>

```



Chant or psalms notation

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

```

stemOff = \hide Staff.Stem
stemOn  = \undo \stemOff

```

```

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

```



Compound time signatures

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

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

```
\relative c' {
  \compoundMeter #'((2 8) (3 8))
  c8 d e fis gis
  c8 fis, gis e d
  c8 d e4 gis8
}
```



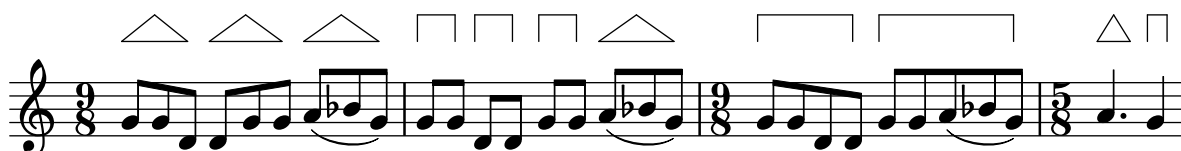
Conducting signs, measure grouping signs

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

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

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

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



Consistently left aligned bar numbers

When left aligning bar numbers, overlapping problems may occur with Staves brackets.

The snippet solves this by keeping right aligned the first bar number following line breaks.

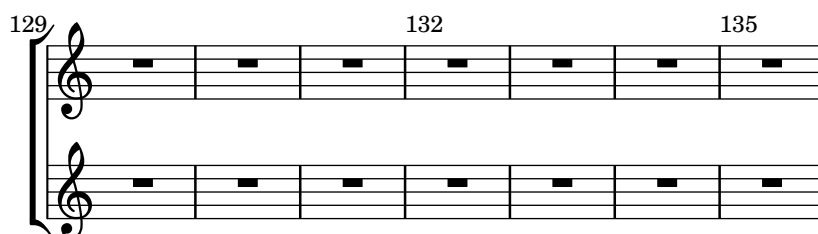
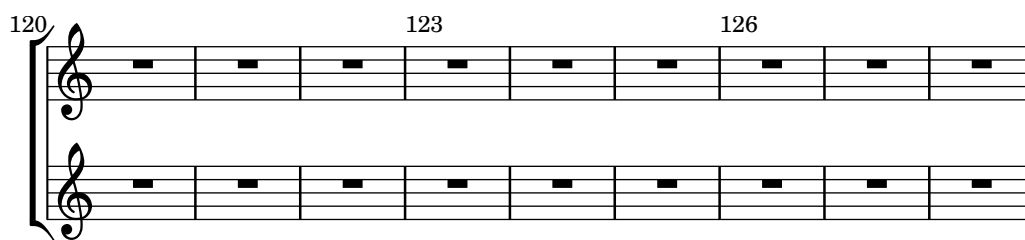
```
consistentlyLeftAlignedBarNumbers = {
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \override Score.BarNumber.self-alignment-X =
    #(lambda (grob)
      (let ((break-dir (ly:item-break-dir grob)))
        (if (= break-dir RIGHT) RIGHT LEFT)))
}

\new ChoirStaff <<
  \new Staff {
    \relative c' {
      \set Score.barNumberVisibility = #(every-nth-bar-number-visible 3)
      \bar ""
      \consistentlyLeftAlignedBarNumbers

      \set Score.currentBarNumber = #112
      \repeat unfold 8 { R1 }
      \break
      \repeat unfold 9 { R1 }
      \break
      \repeat unfold 7 { R1 }
    }
  }
  \new Staff {
    \relative c' {
      \repeat unfold 24 { R1 }
    }
  }
>>

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





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, which is the default behavior), or `#f` (never print a bracket). The latter is in fact equivalent to omitting the `@code{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' {
    << \music s4^"default" >>
    \override TupletBracket.bracket-visibility = #'if-no-beam
    << \music s4^"'if-no-beam" >>
    \override TupletBracket.bracket-visibility = ##t
    << \music s4^"#t" >>
    \override TupletBracket.bracket-visibility = ##f
    << \music s4^"#f" >>
    \omit TupletBracket
    << \music s4^"omit" >>
  }
}
```



Creating metronome marks in markup mode

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

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



Engraving ties manually

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

Note that LilyPond makes a distinction between exact and inexact values for the first number. 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 vertical position without further adjustments.

```
\relative c' {
  <c e g>2~ <c e g>
  \override TieColumn.tie-configuration =
    #'((0.0 . 1) (-2.0 . 1) (-4.0 . 1))
  <c e g>2~ <c e g>
  \override TieColumn.tie-configuration =
    #'((0 . 1) (-2 . 1) (-4 . 1))
  <c e g>2~ <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 }
}
```

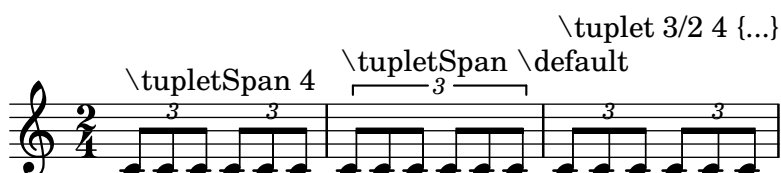


Entering several tuplets using only one \tuplet command

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

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



Flat flags and beam nibs

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

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

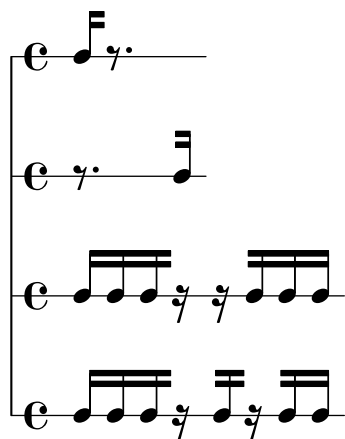
For left-pointing flat flags, set `stemRightBeamCount` instead (Example 2).

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

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

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

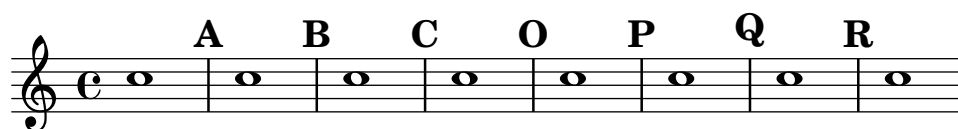
```
\score {
  <<
    % Example 1
    \new RhythmicStaff {
      \set stemLeftBeamCount = #0
      c16[]
      r8.
    }
    % Example 2
    \new RhythmicStaff {
      r8.
      \set stemRightBeamCount = #0
      16[]
    }
    % Example 3
    \new RhythmicStaff {
      16 16
      \set stemRightBeamCount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



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'' {
  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")))
    (ly:stencil-add (ly:font-stencil "d" "u") (ly:stencil-add flag-stencil stroke-stencil))))
```



```

(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) 0)))

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

{
  \override Score.RehearsalMark.self-alignment-X = #LEFT
  \time 1/4
  \mark "Normal flags"
  \snippetexamplenotes

  \mark "Custom flag: inverted"
  \override Flag.stencil = #inverted-flag
  \snippetexamplenotes

  \mark "Custom flag: weight"
  \override Flag.stencil = #weight-flag
  \snippetexamplenotes

  \mark "Revert to normal"
  \revert Flag.stencil
  \snippetexamplenotes
}

```



Guitar strum rhythms

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

```

\include "predefined-guitar-fretboards.ly"
<<
  \new ChordNames {
    \chordmode {
      c1 | f | g | c
    }
  }
}

```

```

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

```

The image displays a musical score for a song. It features two staves: a guitar staff (top) and a melody staff (bottom). The guitar staff shows four chords: C, F, G, and C, each with a fretboard diagram. The melody staff shows a sequence of notes: C4, E4, E4, F4, F4, G4, G4, A4, E4, C4. The lyrics 'This is my song. I like to sing.' are written below the melody staff.

Heavily customized polymetric time signatures

Though the polymetric time signature shown was 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!).

```

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))
}

```

```

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
c4. d4 c4 d4. c4 d c2 d4. e4-^ d4
c4. d4 c4 d4. c4 d c2 d4. e4-^ d4 \break
}

drum = \new DrumStaff \drummode {
  \bar ".|:" bd4.^ \markup { Drums } sn4 bd \bar ";" sn4.
  bd4 sn \bar ";" bd sn bd4. sn4 bd \bar ":|."
}

\new Staff \with {
  instrumentName = \markup { \concat { "B" \flat " Sop." } }
}

{
  \melody
  \drum
}

```

B \flat Sop.

2

4

6

Drums

Making an object invisible with 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
}
>>
}

```



Making slurs with complex dash structure

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

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

```

\relative c' {
  \once \override
    Slur.dash-definition = #'((0 0.3 0.1 0.75)
                              (0.3 0.6 1 1)
                              (0.65 1.0 0.4 0.75))

  c4( d e f)
  \once \override
    Slur.dash-definition = #'((0 0.25 1 1)
                              (0.3 0.7 0.4 0.75)
                              (0.75 1.0 1 1))

  c4( d e f)
}

```



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

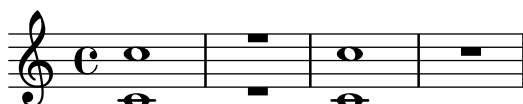
When using multi-measure rests in a polyphonic staff, the rests will be placed differently depending on the voice they belong to. However they can be printed on the same staff line, using the following setting.

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

```

{
  <<
  {
    c''1
    R1
    c''1
    \normalPos
    R1
  }
  \\\
  {
    c'1
    R1
    c'1
    \normalPos
    R1
  }
  >>
}

```

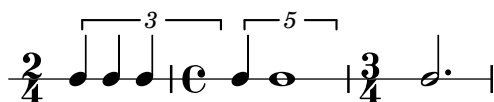


Modifying tuplet bracket length

Tuplet brackets can be made to run to prefatory matter or the next note. 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 {
  % 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.
}
```



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 }
>>
```

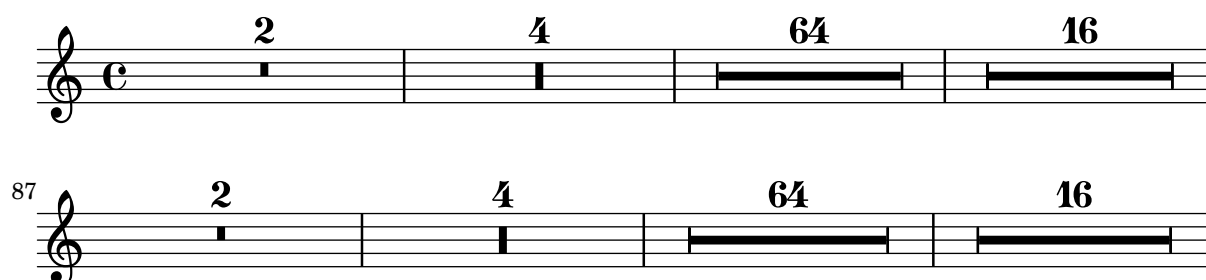


Multi-measure rest length control

Multi-measure rests have length according to their total duration which is under the control of `MultiMeasureRest.space-increment`. Note that the default value is 2.0.

```
\relative c' {
  \compressEmptyMeasures
  R1*2 R1*4 R1*64 R1*16
  \override Staff.MultiMeasureRest.space-increment = 2.5
  R1*2 R1*4 R1*64 R1*16
}
```

}

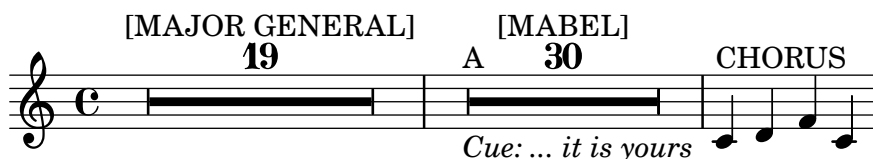


Multi-measure rest markup

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

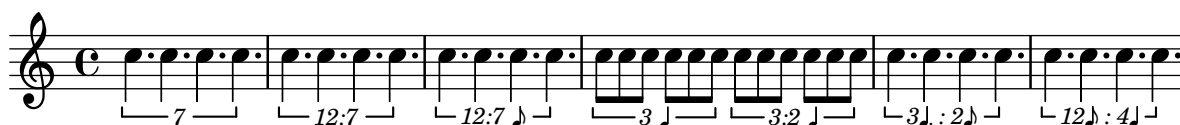
LilyPond also provides formatting functions to print tuplet numbers different than the actual fraction, as well as to append a note value to the tuplet number or tuplet fraction.

```
\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. }
}

```



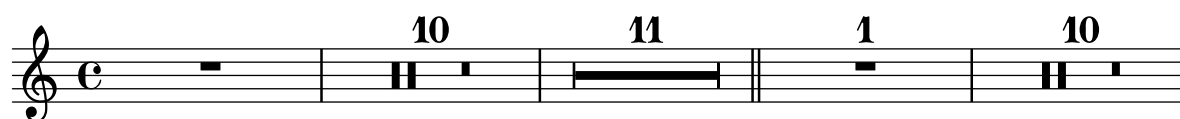
Numbering single measure rests

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 autobeaming will be turned off for the entire staff.

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

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

In order to use `\autoBeamOff` to stop all autobeaming when used with `\partCombine`, it will be necessary to use *three* calls to `\autoBeamOff`.

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



Percussion example

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

```
#(define mydrums '((bassdrum   default #t  4)
                   (snare      default #t -4)
                   (tambourine default #t  0)))
```

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

```
drumsA = {
  \context DrumVoice <<
```

```

{ \global }
{ \drummode {
  \autoBeamOff
  \stemDown sn8 \stemUp tamb s8 |
  sn4 \stemDown sn4 |
  \stemUp tamb8 \stemDown sn8 \stemUp sn16 \stemDown sn \stemUp sn8 |
  \stemDown sn8 \stemUp tamb s8 |
  \stemUp sn4 s8 \stemUp tamb
}
}
>>
}

drumsB = {
  \drummode {
    s4 bd8 s2*2 s4 bd8 s4 bd8 s8
  }
}

\layout {
  indent = #40
}

\score {
  \new StaffGroup <<
    \new DrumStaff \with {
      instrumentName = \markup {
        \center-column {
          "Tambourine"
          "et"
          "caisse claire s. timbre"
        }
      }
    }
    drumStyleTable = #(alist->hash-table mydrums)
  }
  \drumsA
  \new DrumStaff \with {
    instrumentName = #"Grosse Caisse"
    drumStyleTable = #(alist->hash-table mydrums)
  }
  \drumsB
  >>
}

```

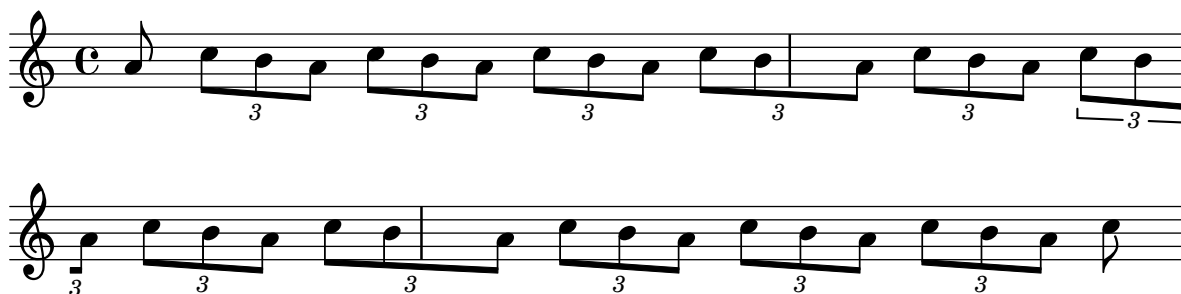
Tambourine
et
caisse claire s. timbre

Grosse Caisse

Permitting line breaks within beamed tuplets

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

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



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.

```
\relative c'' {
  <<
    \override Score.SpacingSpanner.strict-grace-spacing = ##t
    \new Staff \new Voice {
      \afterGrace c4 { c16[ c8 c16] }
      c8[ \grace { b16 d } c8]
      c4 r
    }
    \new Staff {
      c16 c c c c c c c c4 r
    }
  >>
}
```



Positioning multi-measure rests

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

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

  % In two Voices, odd-numbered voices are under the top line
  << { R1 } \ { a1 } >>
  % Even-numbered voices are under the bottom line
```

```

<< { a1 } \\ { R1 } >>
% Multi-measure rests in both voices remain separate
<< { R1 } \\ { R1 } >>

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

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

```



Preventing final mark from removing final tuplet

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 `false`.

```

% due to issue 2362 a long mark such as
% \mark "Composed Feb 2007 - Feb 2008"
% cannot be used here.

```

```

\new Staff {
  \set tupletFullLength = ##t
  \time 1/8
  \tuplet 3/2 8 { c'16 c' c' c' c' c' c' c' }
  \override Score.RehearsalMark.break-visibility = ##(t t t)
  \override Score.RehearsalMark.direction = #DOWN
  \override Score.RehearsalMark.self-alignment-X = #RIGHT
  \mark "1234"
}

```

```

\new Staff {
  \set tupletFullLength = ##t

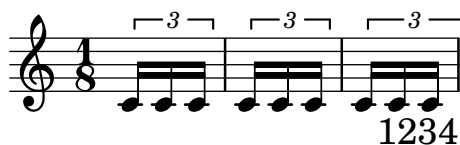
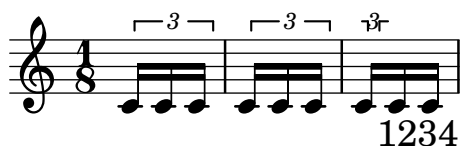
```

```

\override TupletBracket.full-length-to-extent = ##f

\time 1/8
\tuplet 3/2 8 { c'16 c' c' c' c' c' c' c' }
\override Score.RehearsalMark.break-visibility = ##(#t #t #t)
\override Score.RehearsalMark.direction = #DOWN
\override Score.RehearsalMark.self-alignment-X = #RIGHT
\mark "1234"
}

```



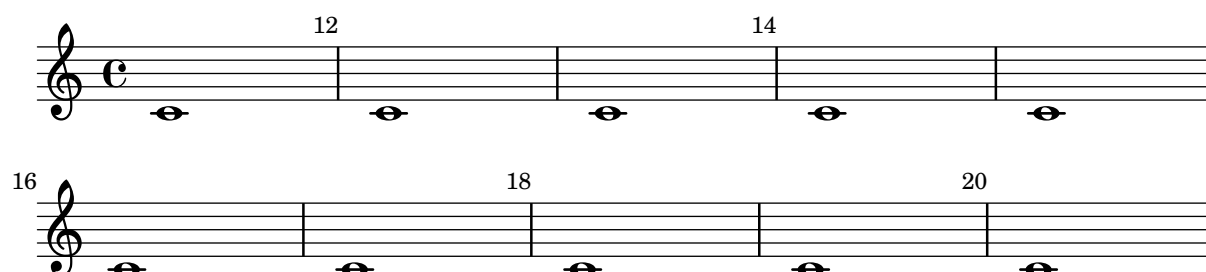
Printing bar numbers at regular intervals

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

```

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

```



Printing bar numbers inside boxes or circles

Bar numbers can also be printed inside boxes or circles.

```

\relative c' {
  % Prevent bar numbers at the end of a line and permit them elsewhere
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 4)
}

```

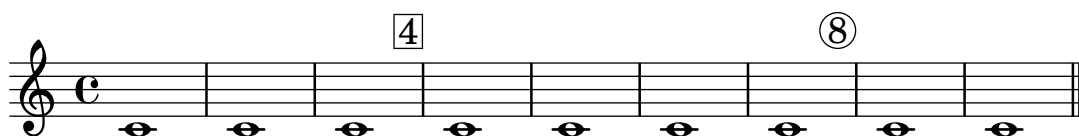
```

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

% Draw a box round the following bar number(s)
\override Score.BarNumber.stencil
  = #(make-stencil-boxer 0.1 0.25 ly:text-interface::print)
\repeat unfold 5 { c1 }

% Draw a circle round the following bar number(s)
\override Score.BarNumber.stencil
  = #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
\repeat unfold 4 { c1 } \bar "|."
}

```



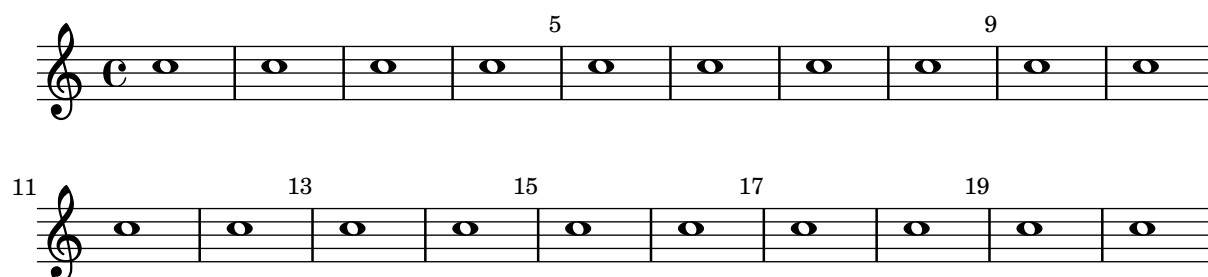
Printing bar numbers with changing regular intervals

The bar number interval can be changed by changing the context function `{set-bar-number-visibility}`.

```

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

```



Printing metronome and rehearsal marks below the staff

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

```

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

{
  % Metronome marks below the staff

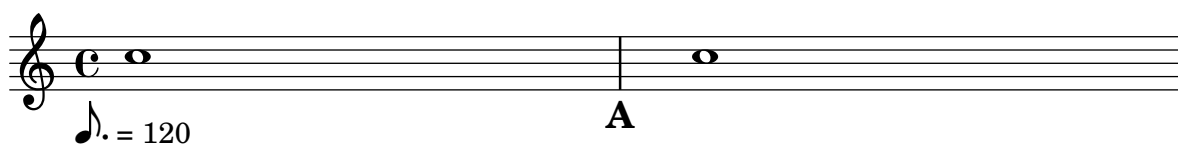
```

```

\override Score.MetronomeMark.direction = #DOWN
\tempo 8. = 120
c''1

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

```



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.

```

\paper {
  indent = #0
  ragged-right = ##t
}

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

\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Default_bar_line_engraver"
    \remove "Bar_number_engraver"
    \override SpacingSpanner.uniform-stretching = ##t
    \override SpacingSpanner.strict-note-spacing = ##t
    proportionalNotationDuration = #(ly:make-moment 1/64)
  }
  \context {
    \Staff
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  }
  \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 {
    \bar "|"
    \clef treble
    \time 3/8
    d''4.

    \bar "|"
    \time 3/4
    r8 des''2( c''8)

    \bar "|"
    \time 7/8
    r4. ees''2 ~

    \bar "|"
    \time 2/4
    \tupletUp
    \tuplet 3/2 { ees''4 r4 d''4 ~ }

    \bar "|"
    \time 3/8
    \tupletUp
    \tuplet 4/3 { d''4 r4 }

    \bar "|"
    \time 2/4
    e''2

    \bar "|"
    \time 3/8
    es''4.

    \bar "|"
    \time 3/4
    r8 d''2 r8
    \bar "|"
  }
>>

Perkussion = \new StaffGroup <<
  \new Staff <<
    \global {
      \bar "|"
      \clef percussion
    }
  >>
  >>

```

```

\time 3/4
r4 c'2 ~

\bar "|"
c'2.

\bar "|"
R2.

\bar "|"
r2 g'4 ~

\bar "|"
g'2. ~

\bar "|"
g'2.
}
>>
\new Staff <<
\global {
\bar "|"
\clef percussion
\time 3/4
R2.

\bar "|"
g'2. ~

\bar "|"
g'2.

\bar "|"
r4 g'2 ~

\bar "|"
g'2 r4

\bar "|"
g'2.
}
>>
>>

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

```

The image displays three systems of musical notation. The first system features a voice line in 3/8 time and piano accompaniment in 3/4 time. The second system, marked with a (4), continues the piece with changing time signatures (3/8, 3/4, 2/4, 3/8, 3/4) and includes triplet and sextuplet markings. The third system, marked with an 8, shows a continuation of the piano accompaniment 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'. By setting `barNumberVisibility` to `all-bar-numbers-visible`, any bar number can be printed for the first measure and all subsequent measures. Note that an empty bar line must be inserted before the first note for this to work.

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

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

The image shows a musical staff in common time (C) with four quarter notes: C, D, E, and F. The bar number '1' is printed above the first measure.



Redefining grace note global defaults

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

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

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

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

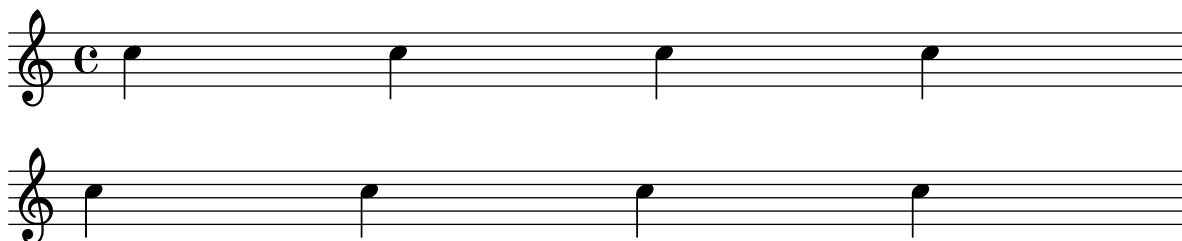


Removing bar numbers from a score

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"
  }
}
```

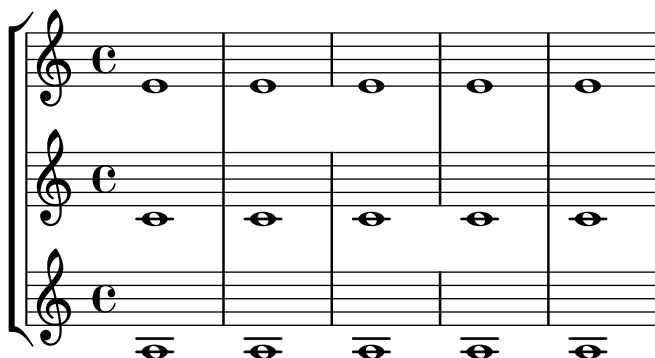
```
\relative c'' {
  c4 c c c \break
  c4 c c c
}
```



Removing connecting bar lines on StaffGroup, PianoStaff, or GrandStaff

By default, bar lines in StaffGroup, PianoStaff, or GrandStaff groups are connected between the staves, i.e. a SpanBar 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
    }
  >>
}
```



Rest styles

Rests may be used in various styles.

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

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

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

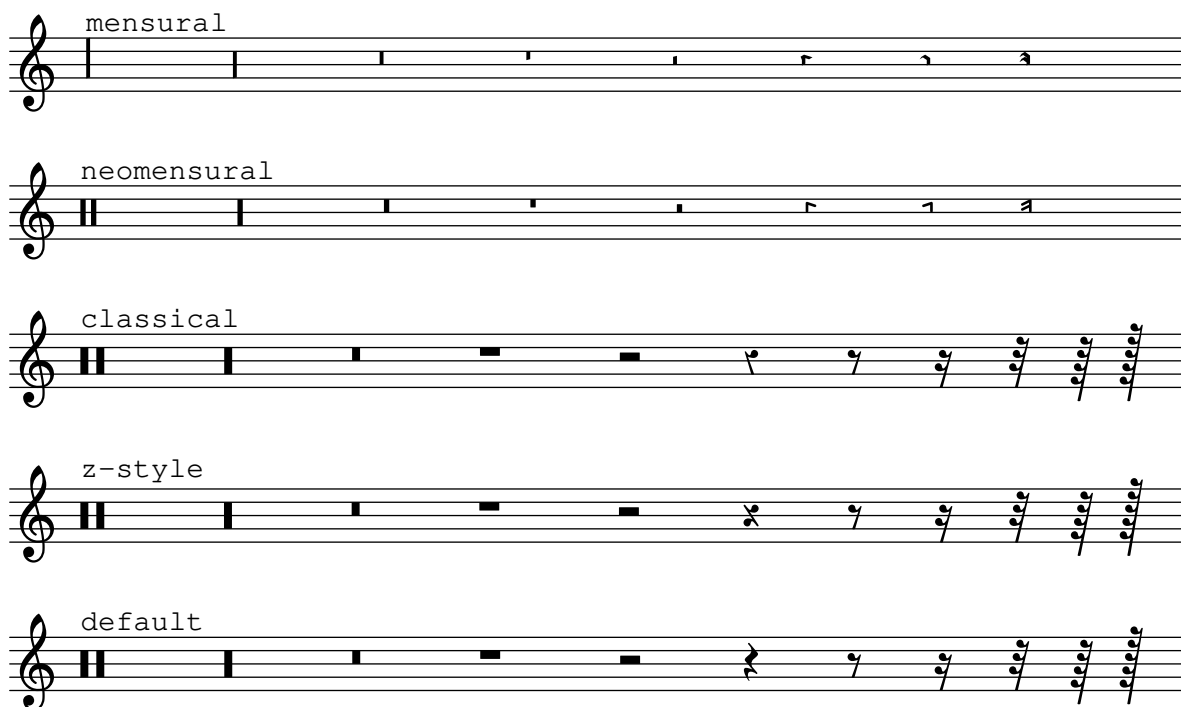
```

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

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

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

```



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 giving the structure of a song. Such a feature is for example useful while creating/transcribing the structure of a song and also when sharing lead sheets with guitarists or jazz musicians.

The standard support for this using `\repeat percent` is unsuitable here since the first beat has to be an ordinary note or rest.

This example shows two solutions to this problem, by redefining ordinary rests to be printed as slashes. (If the duration of each beat is not a quarter note, replace the `r4` in the definitions with a rest of the appropriate duration).

```
% Macro to print single slash
rs = {
  \once \override Rest.stencil = #ly:percent-repeat-item-interface::beat-slash
  \once \override Rest.thickness = #0.48
  \once \override Rest.slope = #1.7
  r4
}

% Function to print a specified number of slashes
comp = #(define-music-function (count) (integer?)
  #{
    \override Rest.stencil = #ly:percent-repeat-item-interface::beat-slash
    \override Rest.thickness = #0.48
    \override Rest.slope = #1.7
    \repeat unfold $count { r4 }
    \revert Rest.stencil
  }
)

\score {
  \relative c' {
    c4 d e f |
    \rs \rs \rs \rs |
    \comp #4 |
  }
}
```



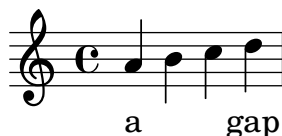
Skips in lyric mode (2)

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.

So for example:

```
<<
\relative c' { a4 b c d }
```

```
\new Lyrics \lyricmode { a4 "" _ gap }
>>
```



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 { \skip 1 bla1 }
>>
```



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`, while rests are hidden by setting `'transparent = ##t`.

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

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

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

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

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

  % Shorten beam by 3/32
  \set baseMoment = #(ly:make-moment 1/8)
  \set beatStructure = 2,2,2,2
```

```

c32[ c c c c] r16.
r2
}

```



Tam-tam example

A tam-tam example, entered with 'tt'

```

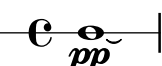
#(define mydrums '((tamtam default #t 0)))

\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 

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))

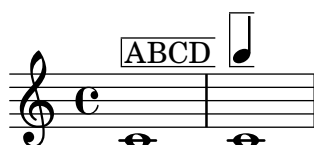
% The corresponding markup command, based on the \box command defined
% in scm/define-markup-commands.scm
#(define-markup-command (NWS-box layout props arg) (markup?)
  #:properties ((thickness 0.1) (font-size 0) (box-padding 0.2))
  "Draw a box round @var{arg}. Looks at @code{thickness},
@code{box-padding}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

% Test it:

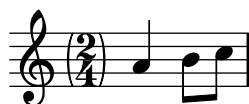
```
\relative c' {
  c1^\markup { \NWS-box ABCD }
  c1^\markup { \NWS-box \note {4} #1.0 }
}
```



Time signature in parentheses - method 3

Another way to put the time signature in parenthesis

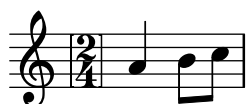
```
\relative c' {
  \override Staff.TimeSignature.stencil = #(lambda (grob)
    (parenthesize-stencil (ly:time-signature::print grob) 0.1 0.4 0.4 0.1 ))
  \time 2/4
  a4 b8 c
}
```



Time signature in parentheses

The time signature can be enclosed within parentheses.

```
\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 printing only the numerator as a number (instead of the fraction)

Sometimes, a time signature should not print the whole fraction (e.g. 7/4), but only the numerator (7 in this case). This can be easily done by using `\override Staff.TimeSignature.style = #'single-digit` to change the style permanently. By using

`\revert Staff.TimeSignature.style`, this setting can be reversed. To apply the single-digit style to only one time signature, use the `\override` command and prefix it with a `\once`.

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



Tweaking grace layout within music

The layout of grace expressions can be changed throughout the music using the functions `add-grace-property` and `remove-grace-property`.

The following example undefines the `Stem` direction for this grace, so that stems do not always point up, and changes the default note heads to crosses.

```
\relative c'' {
  \new Staff {
    $(remove-grace-property 'Voice 'Stem 'direction)
    $(add-grace-property 'Voice 'NoteHead 'style 'cross)
    \new Voice {
      \acciaccatura { f16 } g4
      \grace { d16 e } f4
      \appoggiatura { f,32 g a } e2
    }
  }
}
```



User defined time signatures

New time signature styles can be defined. The time signature in the second measure should be 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 styles of flag on eighth and shorter notes can be displayed by overriding the `stencil` property of `Flag`. Valid values are `modern-straight-flag`, `old-straight-flag` and `flat-flag`.

```
testnotes = {
  \autoBeamOff
  c8 d16 c32 d64 \acciaccatura { c8 } d64 r4
}

\score {
  \relative c' {
    \time 2/4
    \testnotes

    \override Flag.stencil = #modern-straight-flag
    \testnotes

    \override Flag.stencil = #old-straight-flag
    \testnotes

    \override Flag.stencil = #flat-flag
    \testnotes

    \revert Flag.stencil
    \testnotes
  }
}
```

```

}
\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
}

```



Expressive marks

Section “Expressive marks” in *Notation Reference*

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

LilyPond syntax can involve many unusual placements for parentheses, brackets etc., which might sometimes have to be interleaved.

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 parentheses around an expressive mark or chordal note

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

```
\relative c' {
  c2-\parenthesize ->
  \override ParenthesesItem.padding = #0.1
  \override ParenthesesItem.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 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
}

```



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 } >> |

% vee
\override BreathingSign.text = \markup { \musicglyph "scripts.upbow" }
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 postscript code within 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 breathe 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 DynamicText.

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)))
```

```

(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 text) X CENTER))
    X LEFT))
(staff-space (ly:output-def-lookup
  (ly:grob-layout grob) 'staff-space))
(staff-line-thickness
  (ly:output-def-lookup (ly:grob-layout grob) 'line-thickness))
(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))))
#})

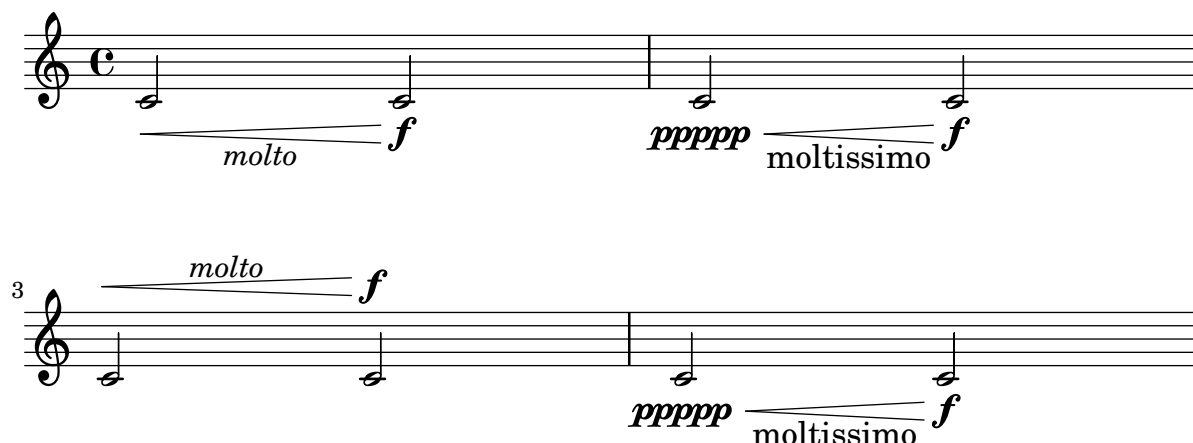
hairpinMolto =
\hairpinWithCenteredText \markup { \italic molto }

hairpinMore =
\hairpinWithCenteredText \markup { \larger moltissimo }

\layout { ragged-right = ##f }

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

```



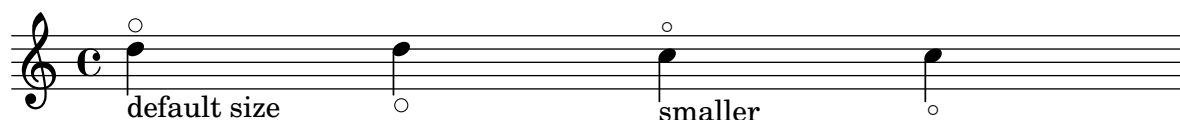
Changing \flageolet mark size

To make the `\flageolet` circle smaller use the following tweak.

```
smallFlageolet = \tweak font-size -3 \flageolet
```

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

```
\relative c'' {
  d4^\flageolet_\markup { default size } d_\flageolet
  c4^\smallFlageolet_\markup { smaller } c_\smallFlageolet
}
```



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 changed independently of the dots placed after the note.

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



Combining dynamics with markup texts

Some dynamics may involve text indications (such as “più forte” or “piano subito”). These can be produced using a `\markup` block.

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

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

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

```
\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 vertical ordering of scripts

The vertical ordering of scripts is controlled with the 'script-priority property. The lower this number, the closer it will be put to the note. In this example, the `TextScript` (the *sharp* symbol) first has the lowest priority, so it is put lowest in the first example. In the second, the *prall trill* (the `Script`) has the lowest, so it is on the inside. When two objects have the same priority, the order in which they are entered determines which one comes first.

```

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

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

```



Creating a delayed turn

Creating a delayed turn, where the lower note of the turn uses the accidental, requires several overrides. The `outside-staff-priority` property must be set to `#f`, as otherwise this would take precedence over the `avoid-slur` property. Changing the fractions `2/3` and `1/3` adjusts the horizontal position.

```

\relative c'' {
  c2*2/3 ( s2*1/3\turn d4) r
  <<

```



```

{ c4.( d8) }
{ s4 s\turn }
>>
\transpose c d \relative c'' <<
{ c4.( d8) }
{
  s4
  \once \set suggestAccidentals = ##t
  \once \override AccidentalSuggestion.outside-staff-priority = ##f
  \once \override AccidentalSuggestion.avoid-slur = #'inside
  \once \override AccidentalSuggestion.font-size = -3
  \once \override AccidentalSuggestion.script-priority = -1
  \single \hideNotes
  b8-\turn \noBeam
  s8
}
>>
}

```



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`.

```

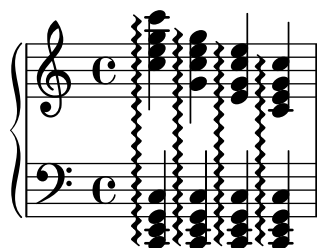
\new PianoStaff \relative c'' <<
  \set PianoStaff.connectArpeggios = ##t
  \new Staff {

```

```

    <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`, `PianoStaff` and `StaffGroup` if the `Span_arpeggio_engraver` is included in the `Score` context.

```

\score {
  \new ChoirStaff {
    \set Score.connectArpeggios = ##t
    <<
      \new Voice \relative c' {
        <c e>2\arpeggio
        <d f>2\arpeggio
        <c e>1\arpeggio
      }
      \new Voice \relative c {
        \clef bass
        <c g'>2\arpeggio
        <b g'>2\arpeggio
        <c g'>1\arpeggio
      }
    >>
  }
  \layout {
    \context {
      \Score
      \consists "Span_arpeggio_engraver"
    }
  }
}

```



Creating double-digit fingerings

Creating fingerings larger than 5 is possible.

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



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 will 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 will be regarded as a dynamic, and therefore will remain 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 simultaneous rehearsal marks

Unlike text scripts, rehearsal marks cannot be stacked at a particular point in a score: only one `RehearsalMark` object is created. Using an invisible measure and bar line, an extra rehearsal mark can be added, giving the appearance of two marks in the same column.

This method may also prove useful for placing rehearsal marks at both the end of one system and the start of the following system.

```
{
  \key a \major
  \set Score.markFormatter = #format-mark-box-letters
  \once \override Score.RehearsalMark.outside-staff-priority = #5000
  \once \override Score.RehearsalMark.self-alignment-X = #LEFT
  \once \override Score.RehearsalMark.break-align-symbols = #'(key-signature)
  \mark \markup { \bold { Senza denti } }

  % the hidden measure and bar line
  % \cadenzaOn turns off automatic calculation of bar numbers
  \cadenzaOn
  \once \override Score.TimeSignature.stencil = ##f
  \time 1/16
  s16 \bar ""
  \cadenzaOff

  \time 4/4
  \once \override Score.RehearsalMark.self-alignment-X = #LEFT
  \mark \markup { \box \bold Intro }
  d'1
  \mark \default
  d'1
}
```



Creating slurs across voices

In some situations, it may be 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 2nd 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

  \set Staff.middleCPosition = #-13
  \override TextSpanner.dash-period = #10
  \override TextSpanner.dash-fraction = #0.5
  \override TextSpanner.thickness = #10
  a4 \startTextSpan
  b4 c
  a4 \stopTextSpan
}

```



Dynamics custom text spanner postfix

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

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

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

```
mydecrec =
#(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-\mydecrec "custom decrec" c4 c4 c4 |
  c4 c4\! c4 c4
}
```



Dynamics text spanner postfix

Custom text spanners can be defined and used with hairpin and text crescendos. `\<` and `\>` produce hairpins by default, `\cresc` etc. produce text spanners by default.

% Some sample text dynamic spanners, to be used as postfix operators
crpoco =

```
#(make-music 'CrescendoEvent
  'span-direction START
  'span-type 'text
  'span-text "cresc. poco a poco")
```

```
\relative c' {
  c4\cresc d4 e4 f4 |
  g4 a4\! b4\crpoco c4 |
  c4 d4 e4 f4 |
  g4 a4\! b4\< c4 |
  g4\dim a4 b4\decrec c4\!
```

}



Glissandi can skip grobs

NoteColumn grobs can be skipped over by glissandi.

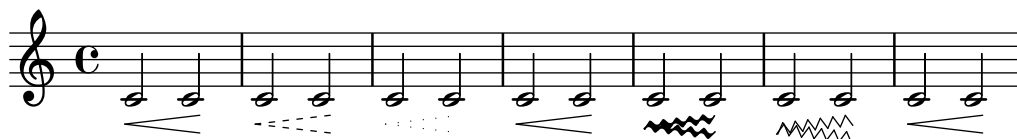
```
\relative c' {
  a2 \glissando
  \once \override NoteColumn.glissando-skip = ##t
  f''4 d,
}
```



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 in the following way:

```
\relative c' {
  \override DynamicTextSpanner.style = #'none
  \crescTextCresc
```

```
c1\< | d | b | c\!
```

```
}
```



Horizontally aligning custom dynamics (e.g. "sempre pp" "piu f" "subito p")

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\overrideX-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: To LilyPond “sempre” has no extent, so it might put other stuff there and create collisions (which are not detected by the collision detection!). Also, there seems to be some spacing, so it’s not exactly the same alignment as without the additional text

- * Add an explicit shifting 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 = 2.5\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 = "s" \with { instrumentName = \markup \column { Normal } }
  <<
    \relative c' {

```

```

        \key es \major
        c4\pp c\p c c | c\ff c c\pp c
    }
>>
\new Staff = "sMarkup" \with {
    instrumentName = \markup \column { Normal markup }
}
<<
    \relative c'' {
        \key es \major
        c4-\semppMarkup c\p c c | c\ff c c-\semppMarkup c
    }
>>
\new Staff = "sK" \with {
    instrumentName = \markup \column { 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 = "sT" \with {
    instrumentName = \markup \column { Right padding }
}
<<
    \relative c'' {
        \key es \major
        c4\semppT c\p c c | c\ff c c\semppT c
    }
>>
\new Staff = "sM" \with {
    instrumentName = \markup \column { Set dimension "to zero" }
}
<<
    \relative c'' {
        \key es \major
        c4\semppM c\p c c | c\ff c c\semppM c
    }
>>
\new Staff = "sG" \with {
    instrumentName = \markup \column { Shift inside dynamics}
}
<<
    \relative c'' {
        \key es \major
        c4\semppG c\p c c | c\ff c c\semppG c
    }

```

```

>>
\new Staff = "sMII" \with {
  instrumentName = \markup \column { Alignment inside dynamics }
}
<<
  \relative c'' {
    \key es \major
    % Setting to ##f (false) gives the same result
    \override DynamicText.X-offset = #0
    c4\semppMII c\p c c | c\ff c c\semppMII c
  }
>>
>>

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

```

Normal	
Normal markup	
Explicit shifting	
Right padding	
Set dimension to zero	
Shift inside dynamics	
Alignment inside dynamics	

How to print two rehearsal marks above and below the same barline (method 1)

This method prints two 'rehearsal marks', one on top of the other. It shifts the lower rehearsal mark below the staff and then adds padding above it in order to place the upper rehearsal mark above the staff.

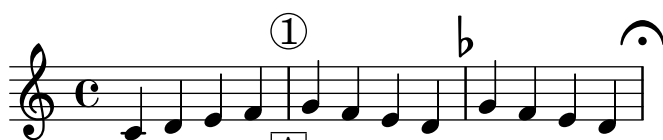
By adjusting the extra-offset and baseline-skip values you can increase or decrease the overall space between the rehearsal mark and the staff.

Because nearly every type of glyph or string can be made to behave like a rehearsal mark it is possible to centre those above and below a bar line.

Adding the appropriate 'break visibility' as shown in snippet 1 (<http://lsr.di.unimi.it/LSR/Item?id=1>) will allow you to position two marks at the end of a line as well.

Note: Method 1 is less complex than Method 2 but does not really allow for fine tuning of placement of one of the rehearsal marks without affecting the other. It may also give some problems with vertical spacing, since using `extra-offset` does not change the bounding box of the mark from its original value.

```
\relative c'{
  c d e f |
  \once \override Score.RehearsalMark.extra-offset = #'(0 . -8.5)
  \once \override Score.RehearsalMark.baseline-skip = #9
  \mark \markup \center-column { \circle 1 \box A }
  g f e d |
  \once \override Score.RehearsalMark.extra-offset = #'(0 . -8.5)
  \once \override Score.RehearsalMark.baseline-skip = #9
  \mark \markup \center-column { \flat { \bold \small \italic Fine. } }
  g f e d |
  \once \override Score.RehearsalMark.extra-offset = #'(0 . -8.5)
  \once \override Score.RehearsalMark.baseline-skip = #9
  \override Score.RehearsalMark.break-visibility = #begin-of-line-invisible
  \mark \markup \center-column { \fermata \box z }
}
```



How to print two rehearsal marks above and below the same barline (method 2)

This method prints two 'rehearsal marks' - one above the staff and one below, by creating two voices, adding the Rehearsal Mark engraver to each voice - without this no rehearsal mark is printed - and then placing each rehearsal mark UP and DOWN in each voice respectively.

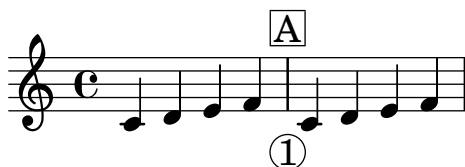
This method (as opposed to method 1) is more complex, but allows for more flexibility, should it be needed to tweak each rehearsal mark independently of the other.

```
\score {
  \relative c'
  <<
  \new Staff {
    <<
    \new Voice \with {
      \consists Mark_engraver
      \consists "Staff_collecting_engraver"
    }
    { c4 d e f
      \mark \markup { \box A }
      c4 d e f
    }
  }
}
```

```

\new Voice \with {
  \consists Mark_engraver
  \consists "Staff_collecting_engraver"
  \override RehearsalMark.direction = #DOWN
}
{ s4 s s s
  \mark \markup { \circle 1 }
  s4 s s s
}
>>
}
>>
\layout {
  \context {
    \Score
    \remove "Mark_engraver"
    \remove "Staff_collecting_engraver"
  }
}
}

```



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 formatting can be tuned using 'tie-configuration.

```

\relative c' {
  <c e g>4\laissezVibrer r <c f g>\laissezVibrer r
}

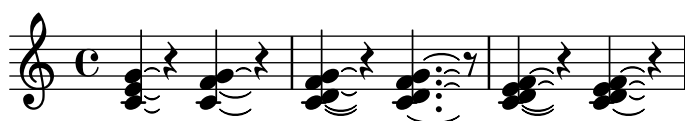
```

```

<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 the Glissando).

```

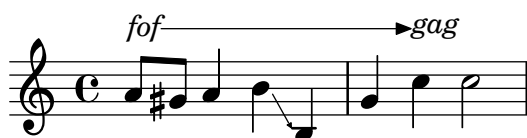
\relative c'' {
  \override TextSpanner.bound-padding = #1.0
  \override TextSpanner.style = #'line
  \override TextSpanner.bound-details.right.arrow = ##t
  \override TextSpanner.bound-details.left.text = #"fof"
  \override TextSpanner.bound-details.right.text = #"gag"
  \override TextSpanner.bound-details.right.padding = #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 made with complex dash patterns by defining the `dash-definition` property. `dash-definition` is a list of `dash-elements`. A `dash-element` is a list of parameters defining the dash behavior for a segment of the slur.

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

fraction dash-fraction of each dash-period black. dash-period is defined in terms of staff spaces. dash-fraction is set to 1 for a solid slur.

```
\relative c' {
  \once \override
    Slur.dash-definition = #'((0 0.3 0.1 0.75)
                               (0.3 0.6 1 1)
                               (0.65 1.0 0.4 0.75))

  c4( d e f)
  \once \override
    Slur.dash-definition = #'((0 0.25 1 1)
                               (0.3 0.7 0.4 0.75)
                               (0.75 1.0 1 1))

  c4( d e f)
}
```



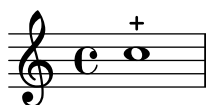
Modifying default values for articulation shorthand notation

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

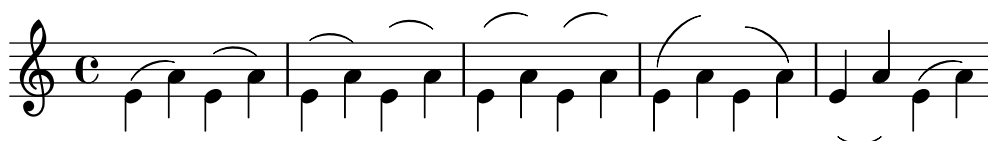
The vertical position of a slur can be adjusted using the `positions` property of `Slur`. The property has 2 parameters, the first referring to the left end of the slur and the second to the right. The values of the parameters are not used by LilyPond to make an exact movement of the slur - instead it 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.

```
\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)
}

```



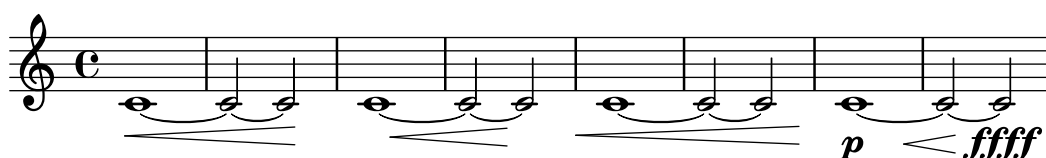
Moving the ends of hairpins

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
}

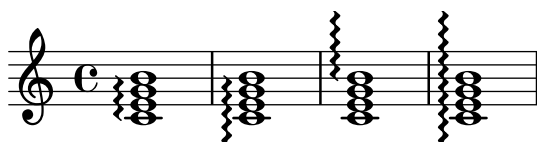
```



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 `false` in order to be printed inside slurs.

```
\relative c'' {
  \override TextScript.avoid-slur = #'inside
  \override TextScript.outside-staff-priority = ##f
  c2(^\markup { \halign #-10 \natural } d4.) c8
}
```



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\s fz\< a a a\!
  \override Hairpin.stencil = #constante-hairpin
  a4\< a a a\f
  a4\p\< a a a\ff
  a4\s fz\< a a a\!
  \override Hairpin.stencil = #flared-hairpin
  a4\> a a a\f
  a4\p\> a a a\ff
  a4\s fz\> a a a\!
  \override Hairpin.stencil = #constante-hairpin
  a4\> a a a\f
}
```

```

a4\p\> a a a\ff
a4\sفز\> 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.

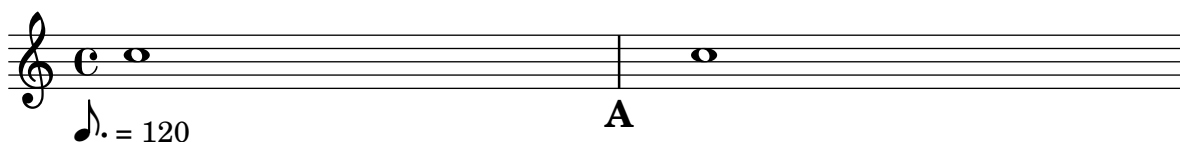
```

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

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

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

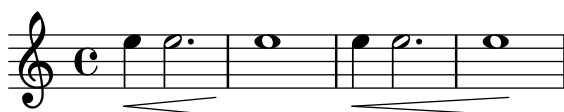
```



Setting hairpin behavior at bar lines

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

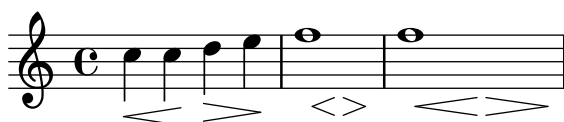
```
\relative c' {
  e4\< e2.
  e1\!
  \override Hairpin.to-barline = ##f
  e4\< e2.
  e1\!
}
```



Setting the minimum length of hairpins

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

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



Showing the same articulation above and below a note or chord

By default, LilyPond does not allow the same articulation (e.g., an accent, a fermata, a flageolet, etc.) to be displayed above and below a note. For example, `c4_fermata^fermata` will only show a fermata below. The fermata above will simply be 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.

So, the solution is to write the note as a chord and add the articulations inside the `<...>`. The direction will always be above, but one can tweak this via a `\tweak`: `<c-\tweak direction #DOWN-\fermata^fermata>`

```
% The same as \flageolet, just a little smaller
smallFlageolet = \tweak font-size #-2 \flageolet
```

```
\relative c' {
  s4^"Wrong:"
  c4\_fermata^fermata % The second fermata is ignored!
  <e d'>4^\smallFlageolet_\smallFlageolet
```

```

s4^"Works if written inside a chord:"
<e_\smallFlageolet d'^\smallFlageolet>4
<e_\flageolet d'^\flageolet>4
<e_\smallFlageolet^\smallFlageolet>4
<e_\fermata^\fermata>4
}

```

Wrong: Works if written inside a chord:



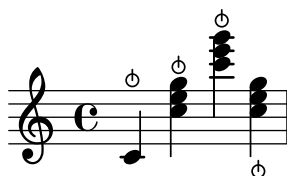
Snap-pizzicato or Bartok pizzicato

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 a tick as the breath mark symbol

Vocal and wind music frequently uses a tick mark as a breathing sign. This indicates a breath that subtracts a little time from the previous note rather than causing a short pause, which is indicated by the comma breath mark. The mark can be moved up a little to take it away from the stave.

```

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

```

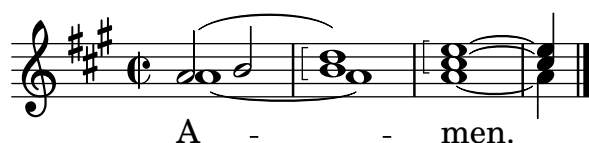


Using arpeggioBracket to make divisi more visible

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

```
\include "english.ly"

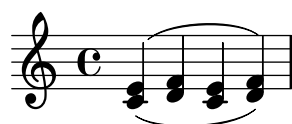
\score {
  \relative c'' {
    \key a \major
    \time 2/2
    <<
      \new Voice = "upper"
      <<
        { \voiceOne \arpeggioBracket
          a2( b2
            <b d>1\arpeggio)
            <cs e>\arpeggio ~
            <cs e>4
          }
        \addlyrics { \lyricmode { A -- men. } }
      >>
      \new Voice = "lower"
      { \voiceTwo
        a1 ~
        a
        a ~
        a4 \bar "|"
      }
    >>
  }
  \layout { ragged-right = ##t }
}
```



Using double slurs for legato chords

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

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



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 by setting the `whiteout` property of `TimeSignature`. To do this `TimeSignature` is moved to a layer above `Tie`, which is left in the default layer of 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
}
```

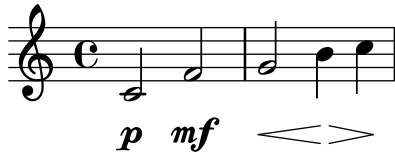


Vertically aligning dynamics across multiple notes

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\!
}



Repeats

Section “Repeats” in *Notation Reference*

Adding volta brackets to additional staves

The `Volta_engraver` by default resides in the `Score` context, and brackets for the repeat are thus normally only printed over the topmost staff. This can be adjusted by adding the `Volta_engraver` to the `Staff` context where the brackets should appear; see also the “Volta multi staff” snippet.

```
<<
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
  \new Staff \with { \consists "Volta_engraver" } { c'2 g' e' a' }
  \new Staff { \repeat volta 2 { c'1 } \alternative { c' } }
>>
```



Centered measure numbers

Scores of large ensemble works often have bar numbers placed beneath the system, centered horizontally on the measure’s extent. This snippet shows how the `Measure_counter_engraver` may be used to simulate this notational practice. Here, the engraver has been added to a `Dynamics` context.

```
\layout {
  \context {
    \Dynamics
    \consists #Measure_counter_engraver
    \override MeasureCounter.direction = #DOWN
    \override MeasureCounter.font-encoding = #'latin1
    \override MeasureCounter.font-shape = #'italic
    % to control the distance of the Dynamics context from the staff:
    \override VerticalAxisGroup.nonstaff-relatedstaff-spacing.padding = #2
  }
  \context {
    \Score
    \remove "Bar_number_engraver"
  }
}
```



```

}

pattern = \repeat unfold 7 { c'4 d' e' f' }

\new StaffGroup <<
  \new Staff {
    \pattern
  }
  \new Staff {
    \pattern
  }
  \new Dynamics {
    \startMeasureCount
    s1*7
    \stopMeasureCount
  }
>>

```



Changing the default bar lines

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

% <http://lsr.di.unimi.it/LSR/Item?id=964>

%%=> <http://lists.gnu.org/archive/html/lilypond-user/2014-03/msg00126.html>

%%=> <http://lilypond.1069038.n5.nabble.com/Changing-the-default-end-repeat-bracket-tc169357>

```

\layout {
  \context {
    \Score
    %% Changing the defaults from engraver-init.ly
    defaultBarType = #"!"
    startRepeatType = #"[:]"
    endRepeatType = #":]"
    doubleRepeatType = #":][[:]"
  }
}

```

%% example:

```

{
  c'1
  \repeat volta 2 { \repeat unfold 2 c' }
  \repeat volta 2 { \repeat unfold 2 c' }
  \alternative {
    { c' }
  }
}

```

```

{
  %% v2.18 workaround
  \once\override Score.VoltaBracket.shorten-pair = #'(1 . -1)
  c'
}
}
\bar "|"
}

```



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 counter

This snippet provides a workaround for emitting measure counters using transparent percent repeats.

```
<<
  \context Voice = "foo" {
```

```

\clef bass
c4 r g r
c4 r g r
c4 r g r
c4 r g r
}
\context Voice = "foo" {
  \set countPercentRepeats = ##t
  \hide PercentRepeat
  \override PercentRepeatCounter.staff-padding = #1
  \repeat percent 4 { s1 }
}
>>

```



Numbering groups of measures

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
  }
}

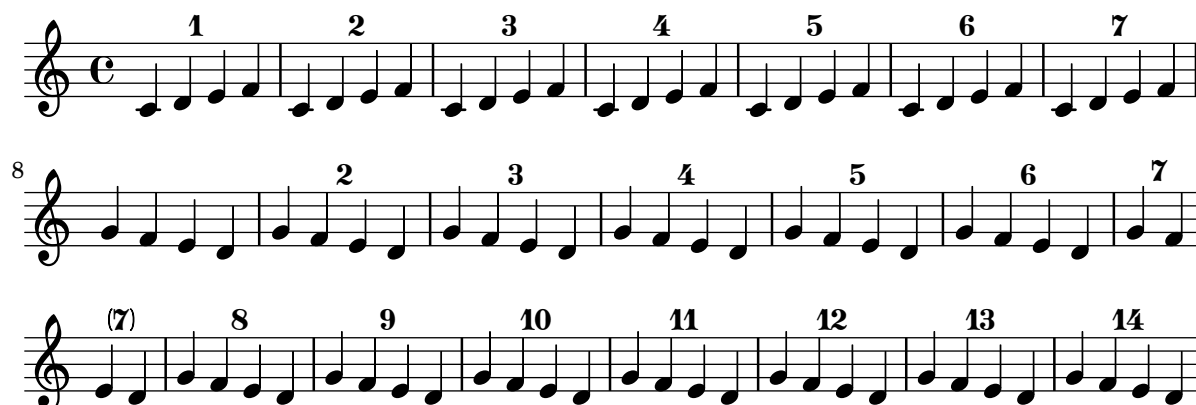
\new Staff {
  \startMeasureCount
  \repeat unfold 7 {
    c'4 d' e' f'
  }
  \stopMeasureCount
  \bar "||"
  g'4 f' e' d'
  \override Staff.MeasureCounter.count-from = #2
  \startMeasureCount
  \repeat unfold 5 {
    g'4 f' e' d'
  }
  g'4 f'
  \bar ""
}

```

```

\break
e'4 d'
\repeat unfold 7 {
  g'4 f' e' d'
}
\stopMeasureCount
}

```



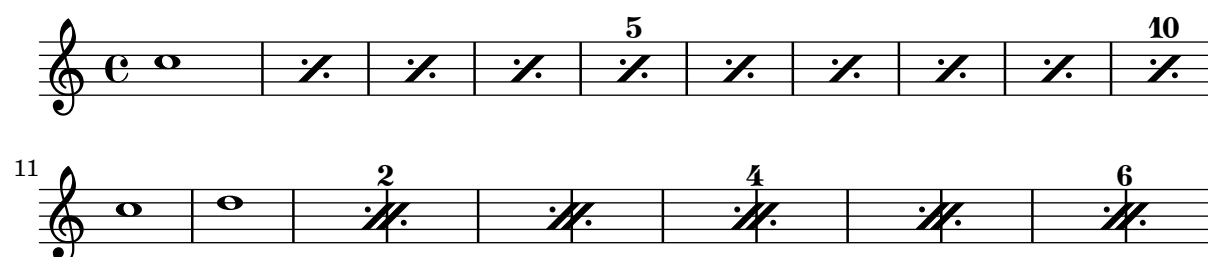
Percent repeat count visibility

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 can get a counter when the convenient property is switched, as shown in this example:

```

\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.

```
{
\clef treble
\key g \major
\time 4/4
\relative c'' {
  \repeat unfold 4 {
    c4 c c c
  }

  % Set segno sign as rehearsal mark and adjust size if needed
  % \once \override Score.RehearsalMark.font-size = #3
  \mark \markup { \musicglyph "scripts.segno" }
  \repeat unfold 2 {
    c4 c c c
  }

  % Set coda sign as rehearsal mark and adjust size if needed
  \once \override Score.RehearsalMark.font-size = #4
  \mark \markup { \musicglyph "scripts.coda" }
  \repeat unfold 2 {
    c4 c c c
  }

  % Should Coda be on anew line?
  % Coda NOT on new line: use \nobreak
  % Coda on new line: DON'T use \nobreak
  % \noBreak

  \bar "||"

  % Set segno sign as rehearsal mark and adjust size if needed
  \once \override Score.RehearsalMark.break-visibility =
    #begin-of-line-invisible
  % \once \override Score.RehearsalMark.font-size = #3
  \mark \markup { \musicglyph "scripts.segno" }

  % Here begins the trickery!
  % \cadenzaOn will suppress the bar count
  % and \stopStaff removes the staff lines.
  \cadenzaOn
  \stopStaff
  % Some examples of possible text-displays

  % text line-aligned
  % =====
  % Move text to the desired position
  % \once \override TextScript.extra-offset = #'( 2 . -3.5 )
```

```

% | <>^\markup { D.S. al Coda } }

% text center-aligned
% =====
% Move text to the desired position
% \once \override TextScript.extra-offset = #'( 6 . -5.0 )
% | <>^\markup { \center-column { D.S. "al Coda" } }

% text and symbols center-aligned
% =====
% Move text to the desired position
% and tweak spacing for optimum text alignment
\repeat unfold 1 {
  s1
  \bar ""
}
\once \override TextScript.extra-offset = #'( 0 . -3.0 )
\once \override TextScript.word-space = #1.5
<>^\markup { \center-column { "D.S. al Coda"
  \line {
    \musicglyph "scripts.coda"
    \musicglyph "scripts.tenuto"
    \musicglyph "scripts.coda"} } }

% Increasing the unfold counter will expand the staff-free space
\repeat unfold 3 {
  s1
  \bar ""
}
% Resume bar count and show staff lines again
\startStaff
\cadenzaOff

% Should Coda be on new line?
% Coda NOT on new line: DON'T use \break
% Coda on new line: use \break
\break

% Show up, you clef and key!
\once \override Staff.KeySignature.break-visibility = #end-of-line-invisible
\once \override Staff.Clef.break-visibility = #end-of-line-invisible

% Set coda sign as rehearsal mark and adjust size and position

% Put the coda sign on top of the (treble-)clef
% depending on coda's line-position

% Coda NOT on new line, use this:
% \once \override Score.RehearsalMark.extra-offset = #'( -2 . 1.75 )

% Coda on new line, use this:
\once \override Score.RehearsalMark.extra-offset = #'( -5 . .5 )

```

```

\once \override Score.RehearsalMark.font-size = #5
\mark \markup { \musicglyph "scripts.coda" }

% The coda
\repeat unfold 6 {
  c4 c c c
}
\bar"|"
}
}

```

D.S. al Coda

⊖ - ⊖

Setting the double repeat default for volte

There are three different styles of double repeats for volte, that can be set using `doubleRepeatType`.

```

\relative c' {
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":...:"
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":|.|:"
  \repeat volta 1 { c1 }
  \set Score.doubleRepeatType = #":|.: "
  \repeat volta 1 { c1 }
}

```

Shortening volta brackets

By default, the volta brackets will be drawn over all of the alternative music, but it is possible to shorten them by setting `voltaSpannerDuration`. In the next example, the bracket only lasts one measure, which is a duration of 3/4.

```

\relative c' {

```



```

\time 3/4
c4 c c
\set Score.voltaSpannerDuration = #(ly:make-moment 3/4)
\repeat volta 5 { d4 d d }
\alternative {
  {
    e4 e e
    f4 f f
  }
  { g4 g g }
}

```



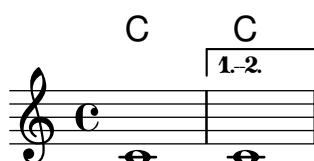
Volta below chords

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

```

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

```



Volta multi staff

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

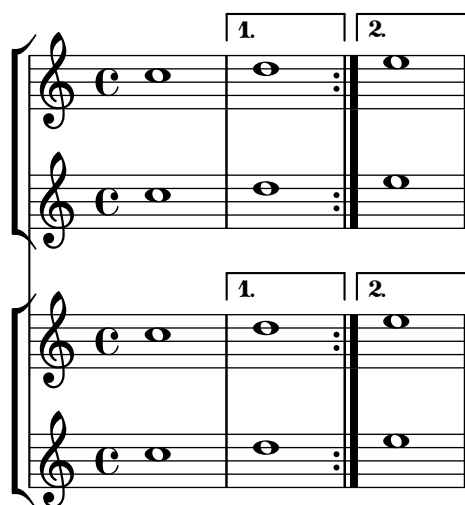
```
voltaMusic = \relative c'' {
```

```

\repeat volta 2 {
  c1
}
\alternative {
  d1
  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 { 1. 2. 3... \text \italic { ad lib. } }
```

```

\relative c'' {
  c1
  \set Score.repeatCommands = #(list (list 'volta voltaAdLib) 'start-repeat)
  c4 b d e
  \set Score.repeatCommands = #'((volta #f) (volta "4.") end-repeat)
}

```

```
f1
\set Score.repeatCommands = #'((volta #f))
}
```



Simultaneous notes

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
    }
  >>
}
```

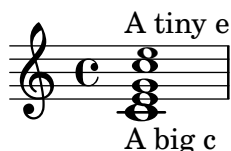


Changing a single note's size in a chord

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 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
  }
}>>
```



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 (`\partCombine` command) allows the combination of several different parts on the same staff. Text directions such as “solo” or “a2” are added by default; to remove them, simply set the property `printPartCombineTexts` to `f`. For vocal scores (hymns), there is no need to add “solo/a2” texts, so they should be switched off. However, it might be better not to use it if there are any solos, as they won’t be indicated. In such cases, standard polyphonic notation may be preferable.

This snippet presents the three ways two parts can be printed on a same staff: standard polyphony, `\partCombine` without texts, and `\partCombine` with texts.

`%% Combining pedal notes with clef changes`

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

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

\score {
  <<
  \new Staff \with { instrumentName = "Standard polyphony" }

  << \musicUp \\\musicDown >>

  \new Staff \with {
    instrumentName = "PartCombine without text"
    printPartCombineTexts = ##f
  }

  \partCombine \musicUp \musicDown

  \new Staff \with { instrumentName = "PartCombine with text" }
  \partCombine \musicUp \musicDown
  >>
\layout {
  indent = 6.0\cm
  \context {
    \Score
    \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 = #11
}

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 with 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
  }
  >>
}

```



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**’ will appear 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 Voices named “two” “one” “solo” and “chords” depending on when and how the parts merged into a common voice. The voices output from **\partCombine** can have their layout properties adjusted in the usual way. Here we define extensions of **\partCombine** to make it easier to put four voices on a staff.

```

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'
  \partCombineUp \soprano \alto
  \partCombineDown \tenor \bass
>>

```

>>

```

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

```

```

customPartCombineUp =
#(define-music-function (partOne partTwo)
  (ly:music? ly:music?)
  "Take the music in @var{partOne} and @var{partTwo} and return
  a @code{VoiceBox} named @q{Up} containing @code{Voice}s
  that contain @var{partOne} and @var{partTwo} merged into one
  voice where feasible. This variant sets the default voicing
  in the output to use upward stems."
#{
  \new VoiceBox = "Up" <<
    \context Voice = "one" { \voiceOne }
    \context Voice = "two" { \voiceThree }
    \context Voice = "shared" { \voiceOne }
    \context Voice = "solo" { \voiceOne }
    \context NullVoice = "null" {}
    \partCombine #partOne #partTwo
  >>
#})

```

```

customPartCombineDown = #
(define-music-function (partOne partTwo)
  (ly:music? ly:music?)
  "Take the music in @var{partOne} and @var{partTwo} and return
  a @code{VoiceBox} named @q{Down} containing @code{Voice}s
  that contain @var{partOne} and @var{partTwo} merged into one
  voice where feasible. This variant sets the default voicing
  in the output to use downward stems."
#{
  \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" {}
  >>
#})

```

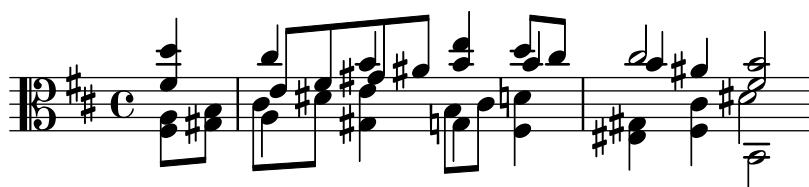
```

\partCombine #partOne #partTwo
>>
#})

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
>>

```



Staff notation

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 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 add a setting of `Staff.explicitKeySignatureVisibility` 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 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 indicators to staves which get split after a break

This snippet defines the `\splitStaffBarLine`, `convUpStaffBarLine` and `convDownStaffBarLine` commands. 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.

```
#(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 {
  ragged-right = ##t
  short-indent = 10\mm
}

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
}

\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"
  } <<

```

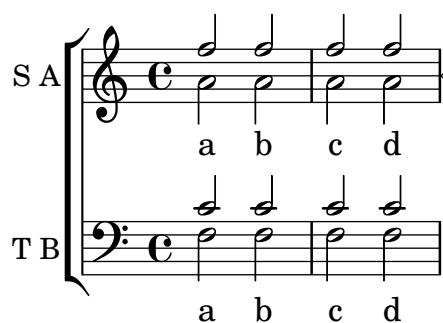


```

\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 shows two musical staves with lyrics. The first staff has three systems of notes (e, f, g, h) with labels SI SII, AI AII, and T B. The second staff has two systems of notes (i, j, k, l) with labels S A and T B.

Adding orchestral cues to a vocal score

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

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

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

clarinet = \relative c' {
```

```

\transposition bes
fis4 d d c
}
\addQuote "clarinet" { \clarinet }

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

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

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

```

The image shows a musical score for a song. The top staff is a vocal line in treble clef with a common time signature. The lyrics "here's the lyr-ics" are written below the notes. The bottom two staves are for piano accompaniment. The right hand (treble clef) is labeled "Flute" and the left hand (bass clef) is labeled "Clar.". The piano part consists of chords and single notes in both hands.

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 |
}
```



Alternative bar numbering

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

```
\relative c' {
  \set Score.alternativeNumberingStyle = #'numbers
  \repeat volta 3 { c4 d e f | }
  \alternative {
    { c4 d e f | c2 d \break }
    { f4 g a b | f4 g a b | f2 a | \break }
  }
```

```

      { c4 d e f | c2 d }
    }
c1 \break
\set Score.alternativeNumberingStyle = #'numbers-with-letters
\repeat volta 3 { c,4 d e f | }
  \alternative {
    { c4 d e f | c2 d \break }
    { f4 g a b | f4 g a b | f2 a | \break }
    { c4 d e f | c2 d }
  }
c1
}

```

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 Section “Graphical Object Interfaces” in *Internals Reference* 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
}
```



Centered measure numbers

Scores of large ensemble works often have bar numbers placed beneath the system, centered horizontally on the measure's extent. This snippet shows how the `Measure_counter_engraver` may be used to simulate this notational practice. Here, the engraver has been added to a `Dynamics` context.

```
\layout {
  \context {
    \Dynamics
    \consists #Measure_counter_engraver
    \override MeasureCounter.direction = #DOWN
    \override MeasureCounter.font-encoding = #'latin1
    \override MeasureCounter.font-shape = #'italic
    % to control the distance of the Dynamics context from the staff:
    \override VerticalAxisGroup.nonstaff-relatedstaff-spacing.padding = #2
  }
  \context {
    \Score
    \remove "Bar_number_engraver"
  }
}
```

```
pattern = \repeat unfold 7 { c'4 d' e' f' }
```

```
\new StaffGroup <<
  \new Staff {
    \pattern
  }
  \new Staff {
    \pattern
  }
  \new Dynamics {
    \startMeasureCount
    s1*7
    \stopMeasureCount
  }
>>
```



Changing the default bar lines

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

```
% http://lsr.di.unimi.it/LSR/Item?id=964
%%=> http://lists.gnu.org/archive/html/lilypond-user/2014-03/msg00126.html
%%=> http://lilypond.1069038.n5.nabble.com/Changing-the-default-end-repeat-bracket-tc169357

\layout {
  \context {
    \Score
    %% Changing the defaults from engraver-init.ly
    defaultBarType = #"!"
    startRepeatType = #"[:]"
    endRepeatType = #":|]"
    doubleRepeatType = #":||[:]"
  }
}

%% example:
{
  c'1
  \repeat volta 2 { \repeat unfold 2 c' }
  \repeat volta 2 { \repeat unfold 2 c' }
  \alternative {
    { c' }
    {
      %% v2.18 workaround
      \once\override Score.VoltaBracket.shorten-pair = #'(1 . -1)
      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 xx}`, 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 20)

```



```

\score {
  {
    \repeat unfold 12 { s1 \break }
  }
  \layout {
    indent = 0\in
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Clef_engraver"
      \remove "Bar_engraver"
    }
    \context {
      \Score
      \remove "Bar_number_engraver"
    }
  }
}

% 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
}
%}

% uncomment these lines for "A4" size
%{
\paper {
  #(set-paper-size "a4")
  ragged-last-bottom = ##f
  line-width = 180
  left-margin = 15
  bottom-margin = 10
  top-margin = 10
}
%}

```





Creating custom key signatures

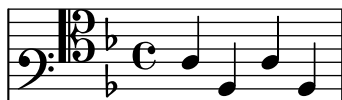
LilyPond supports custom key signatures. In this example, print for D minor with an extended range of printed flats.

```
\new Staff \with {
  \override StaffSymbol.line-count = #8
  \override KeySignature.flat-positions = #'((-7 . 6))
  \override KeyCancellation.flat-positions = #'((-7 . 6))
  % presumably sharps are also printed in both octaves
  \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
```

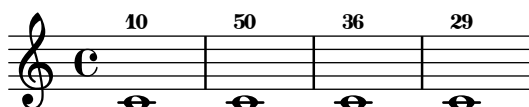
```
f bes, f bes,
}
```



Creating double-digit fingerings

Creating fingerings larger than 5 is possible.

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



Cross staff stems

This snippet shows the use of the `Span_stem_engraver` and `\crossStaff` to connect stems across staves automatically.

The stem length need not be specified, as the variable distance between noteheads and staves is calculated automatically.

```
\layout {
  \context {
    \PianoStaff
    \consists #Span_stem_engraver
  }
}

{
  \new PianoStaff <<
  \new Staff {
    <b d'>4 r d'16\> e'8. g8 r\!
    e'8 f' g'4 e'2
  }
  \new Staff {
    \clef bass
    \voiceOne
    \autoBeamOff
    \crossStaff { <e g>4 e, g16 a8. c8} d
    \autoBeamOn
    g8 f g4 c2
  }
  >>
}
```

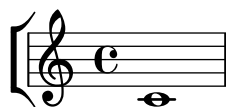


Display bracket with only one staff in a system

If there is only one staff in one of the staff types `ChoirStaff` or `StaffGroup`, by default the bracket and the starting bar line will not be displayed. This can be changed by overriding `collapse-height` to set its value to be less than the number of staff lines in the staff.

Note that in contexts such as `PianoStaff` and `GrandStaff` where the systems begin with a brace instead of a bracket, another property has to be set, as shown on the second system in the example.

```
\score {
  \new StaffGroup <<
    % Must be lower than the actual number of staff lines
    \override StaffGroup.SystemStartBracket.collapse-height = #4
    \override Score.SystemStartBar.collapse-height = #4
    \new Staff {
      c'1
    }
  >>
}
\score {
  \new PianoStaff <<
    \override PianoStaff.SystemStartBrace.collapse-height = #4
    \override Score.SystemStartBar.collapse-height = #4
    \new Staff {
      c'1
    }
  >>
}
```



Extending a TrillSpanner

For `TrillSpanner`, the `minimum-length` property becomes effective only if the `set-spacing-rods` procedure is called explicitly.

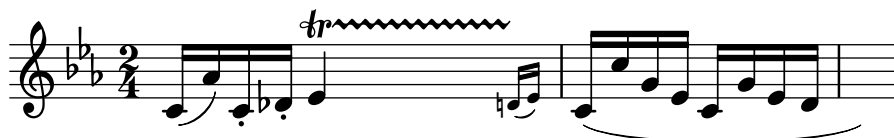
To do this, the `springs-and-rods` property should be set to `ly:spanner::set-spacing-rods`.

```
\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
\hideNotes
c8)
}

```



Extending glissandi across repeats

A glissando which extends into several `\alternative` blocks can be simulated by adding a hidden grace note with a glissando at the start of each `\alternative` block. The grace note should be at the same pitch as the note which starts the initial glissando. This is implemented here with a music function which takes the pitch of the grace note as its argument.

Note that in polyphonic music the grace note must be matched with corresponding grace notes in all other voices.

```
repeatGliss = #(define-music-function (grace)
  (ly:pitch?)
  #{
    % the next two lines ensure the glissando is long enough
    % to be visible
    \once \override Glissando.springs-and-rods
      = #ly:spanner::set-spacing-rods
    \once \override Glissando.minimum-length = #3.5
    \once \hideNotes
    \grace $grace \glissando
  #})
```

```
\score {
  \relative c'' {
    \repeat volta 3 { c4 d e f\glissando }
    \alternative {
      { g2 d }
      { \repeatGliss f g2 e }
      { \repeatGliss f e2 d }
    }
  }
}
```

```
music = \relative c' {
  \voiceOne
  \repeat volta 2 {
    g a b c\glissando
  }
  \alternative {
    { d1 }
  }
}
```

```

    { \repeatGliss c \once \omit StringNumber e1\2 }
  }
}

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

```

The image displays two musical staves. The top staff is a treble clef staff with a C-clef (G8) and a C-clef (G8). It contains three measures of music, each with a first ending bracket labeled 1, 2, and 3. The bottom staff is a tablature staff with a treble clef and a C-clef (G8). It contains two measures of music, each with a first ending bracket labeled 1 and 2. The tablature staff includes fingerings (0, 2, 0, 1, 3, 5) and a repeat sign.

Flat Ties

The function takes the default `Tie.stencil` as an argument, calculating the result relying on the extents of this default.

Further tweaking is possible by overriding `Tie.details.height-limit` or with `\shape`. It's also possible to change the custom-definition on the fly.

%% <http://lsr.di.unimi.it/LSR/Item?id=1031>

```

#(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)
        ;(coord-scale coord (cons x (* y dir)))
        (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))

;; outer let to trigger suicide
(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))))
        (uplist
         (map pair-to-list
              (normalize-coords coords lenx (* leny 2) dir))))

      (ly:stencil-translate
       (my-c-p-s uplist used-thick)
       (cons xtrans ytrans)))
  '()))

#(define flare-tie
  (flared-tie '((0 . 0)(0.1 . 0.2) (0.9 . 0.2) (1.0 . 0.0))))

\layout {
  \context {
    \Voice
    \override Tie.stencil = #flare-tie
  }
}

\paper { ragged-right = ##f }

\relative c' {
  a4~a
  \override Tie.height-limit = 4
  a'4~a
  a'4~a
  <a,, c e a c e a c e>~ q

  \break

  a'4~a
  \once \override Tie.details.height-limit = 14

```

a4~a

\break

a4~a

\once \override Tie.details.height-limit = 0.5

a4~a

\break

a4~a

\shape #'((0 . 0) (0 . 0.4) (0 . 0.4) (0 . 0)) Tie

a4~a

\break

a4~a

\once \override Tie.stencil =

 #(flared-tie '((0 . 0)(0.1 . 0.4) (0.9 . 0.4) (1.0 . 0.0)))

a4~a

a4~a

\once \override Tie.stencil =

 #(flared-tie '((0 . 0)(0.06 . 0.1) (0.94 . 0.1) (1.0 . 0.0)))

a4~a

}

Forcing measure width to adapt to MetronomeMark'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 = {
  \tempo "Allegro"
  R1*6
  \tempo "Rall."
  R1*2
  \tempo "A tempo"
  R1*8
}

{
  \compressMMRests {
    \example
    R1
    R1
    \override Score.MetronomeMark.extra-spacing-width = #'(-3 . 0)
    \example
  }
}
```

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,
}
```

How to print two rehearsal marks above and below the same barline (method 1)

This method prints two 'rehearsal marks', one on top of the other. It shifts the lower rehearsal mark below the staff and then adds padding above it in order to place the upper rehearsal mark above the staff.

By adjusting the extra-offset and baseline-skip values you can increase or decrease the overall space between the rehearsal mark and the staff.

Because nearly every type of glyph or string can be made to behave like a rehearsal mark it is possible to centre those above and below a bar line.

Adding the appropriate 'break visibility' as shown in snippet 1 (<http://lsr.di.unimi.it/LSR/Item?id=1>) will allow you to position two marks at the end of a line as well.

Note: Method 1 is less complex than Method 2 but does not really allow for fine tuning of placement of one of the rehearsal marks without affecting the other. It may also give some problems with vertical spacing, since using `extra-offset` does not change the bounding box of the mark from its original value.

```
\relative c'{
  c d e f |
  \once \override Score.RehearsalMark.extra-offset = #'(0 . -8.5)
  \once \override Score.RehearsalMark.baseline-skip = #9
  \mark \markup \center-column { \circle 1 \box A }
  g f e d |
  \once \override Score.RehearsalMark.extra-offset = #'(0 . -8.5)
  \once \override Score.RehearsalMark.baseline-skip = #9
  \mark \markup \center-column { \flat { \bold \small \italic Fine. } }
  g f e d |
  \once \override Score.RehearsalMark.extra-offset = #'(0 . -8.5)
  \once \override Score.RehearsalMark.baseline-skip = #9
  \override Score.RehearsalMark.break-visibility = #begin-of-line-invisible
  \mark \markup \center-column { \fermata \box z }
}
```



How to print two rehearsal marks above and below the same barline (method 2)

This method prints two 'rehearsal marks' - one above the staff and one below, by creating two voices, adding the Rehearsal Mark engraver to each voice - without this no rehearsal mark is printed - and then placing each rehearsal mark UP and DOWN in each voice respectively.

This method (as opposed to method 1) is more complex, but allows for more flexibility, should it be needed to tweak each rehearsal mark independently of the other.

```
\score {
  \relative c'
  <<
    \new Staff {
      <<
        \new Voice \with {
          \consists Mark_engraver
          \consists "Staff_collecting_engraver"
        }
      }
    }
}
```

```

    { c4 d e f
      \mark \markup { \box A }
      c4 d e f
    }
    \new Voice \with {
      \consists Mark_engraver
      \consists "Staff_collecting_engraver"
      \override RehearsalMark.direction = #DOWN
    }
    { s4 s s s
      \mark \markup { \circle 1 }
      s4 s s s
    }
  >>
}
>>
\layout {
  \context {
    \Score
    \remove "Mark_engraver"
    \remove "Staff_collecting_engraver"
  }
}
}

```



Incipit

When transcribing mensural music, an incipit at the beginning of the piece is useful to indicate the original key and tempo. Musicians today are used to bar lines, but these were not known during the period of mensural music. As a compromise, bar lines are often printed between the staves, a layout style called mensurstriche layout.

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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 = {
  \clef "neomensural-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 = {
  \clef "neomensural-c3"
  \key f \major
  \time 2/2
  r1 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 = {
  \clef "neomensural-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 = {
  \clef "mensural-f"
  \key f \major
  \time 2/2
  r\maxima
  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 \discantusIncipit
        \global
        \discantusNotes
      >>
      \new Lyrics \lyricsto discantusNotes { \discantusLyrics }
      \new Voice = "altusNotes" <<
        \set Staff.instrumentName = "Altus"
        \global
        \incipit \altusIncipit
        \altusNotes
      >>
      \new Lyrics \lyricsto altusNotes { \altusLyrics }
      \new Voice = "tenorNotes" <<
        \set Staff.instrumentName = "Tenor"
        \global
        \incipit \tenorIncipit
        \tenorNotes
      >>
      \new Lyrics \lyricsto tenorNotes { \tenorLyrics }
      \new Voice = "bassusNotes" <<
        \set Staff.instrumentName = "Bassus"
        \global
        \incipit \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 = 6\cm
incipit-width = 4\cm
}
}

```

Discantus

Altus

Tenor

Bassus

Ju - bi - la - te De -

3

o, om - nis ter - ra, om - ... -us.

De - o, om - nis ter - ra, ... -us.

Ju - bi - la - te ... -us.

Ju - bi - ... -us.

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 { ragged-right = ##t  indent = 0\cm }
  }
}

\header {
  title = "Solo Cello Suites"
  subtitle = "Suite IV"
  subsubtitle = \markup { Originalstimmung: \raise #0.5 \tuning }
}

\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 usually the first string is printed topmost. If you want to have it at the bottom change the `stringOneTopmost`-context-property. For a context-wide setting this could be done in `layout` 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.

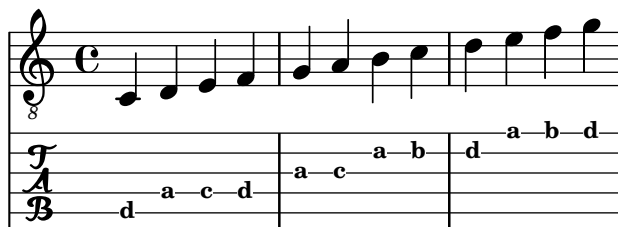
```

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
  }
>>

```

```
}
>>
```

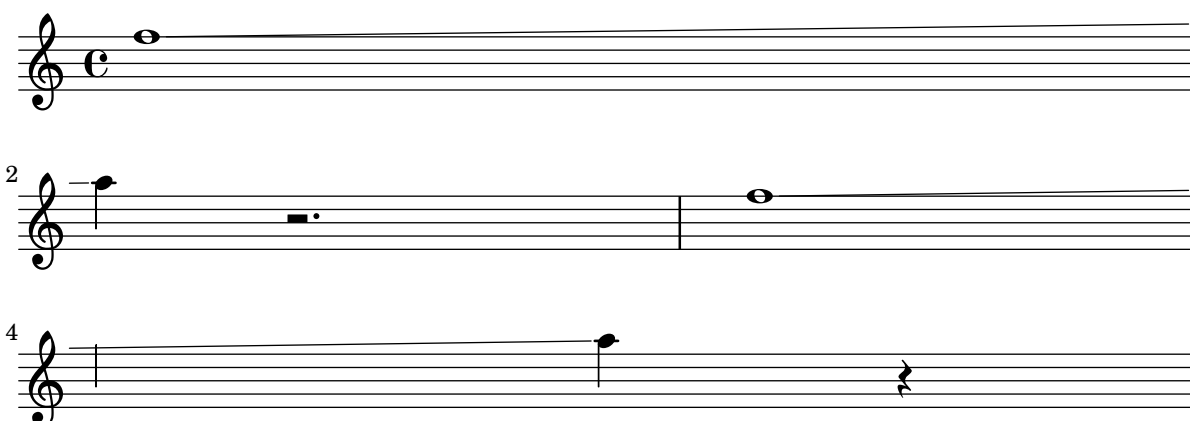


Making glissandi breakable

Setting the `breakable` property to `##t` in combination with `after-line-breaking` allows a glissando to break if it occurs at a line break:

```
glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}

\relative c' {
  \override Glissando.breakable = ##t
  \override Glissando.after-line-breaking = ##t
  f1\glissando |
  \break
  a4 r2. |
  f1\glissando
  \once \glissandoSkipOn
  \break
  a2 a4 r4 |
}
```



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 counter

This snippet provides a workaround for emitting measure counters using transparent percent repeats.

```
<<
  \context Voice = "foo" {
    \clef bass
    c4 r g r
    c4 r g r
    c4 r g r
    c4 r g r
  }
  \context Voice = "foo" {
    \set countPercentRepeats = ##t
    \hide PercentRepeat
    \override PercentRepeatCounter.staff-padding = #1
    \repeat percent 4 { s1 }
  }
>>
```



Mensurstriche layout (bar lines between the staves)

The mensurstriche-layout where the bar lines do not show on the staves but between staves can be achieved with a `StaffGroup` instead of a `ChoirStaff`. The bar line on staves is blanked out using `\hide`.

```
global = {
  \hide Staff.BarLine
  s1 s
  % the final bar line is not interrupted
  \undo \hide Staff.BarLine
  \bar "||."
}

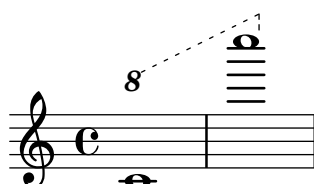
\new StaffGroup \relative c'' {
  <<
    \new Staff { << \global { c1 c } >> }
    \new Staff { << \global { c c } >> }
  >>
}
```



Modifying the Ottava spanner slope

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

```
\relative c'' {
  \override Staff.OttavaBracket.stencil = #ly:line-spanner::print
  \override Staff.OttavaBracket.bound-details =
    #`((left . ((Y . 0) ; Change the integer here
      (attach-dir . ,LEFT)
      (padding . 0)
      (stencil-align-dir-y . ,CENTER)))
      (right . ((Y . 5) ; Change the integer here
        (padding . 0)
        (attach-dir . ,RIGHT)
        (text . ,(make-draw-dashed-line-markup
          (cons 0 -1.2))))))
  \override Staff.OttavaBracket.left-bound-info =
    #ly:line-spanner::calc-left-bound-info-and-text
  \override Staff.OttavaBracket.right-bound-info =
    #ly:line-spanner::calc-right-bound-info
  \ottava #1
  c1
  c'''1
}
```



Nesting staves

The property `systemStartDelimiterHierarchy` can be used to make more complex nested staff groups. The command `\set StaffGroup.systemStartDelimiterHierarchy` takes an alphabetical list of the number of staves produced. Before each staff a system start delimiter can be given. It has to be enclosed in brackets and takes as much staves as the brackets enclose. Elements in the list can be omitted, but the first bracket takes always the complete number of staves. The possibilities are `SystemStartBar`, `SystemStartBracket`, `SystemStartBrace`, and `SystemStartSquare`.

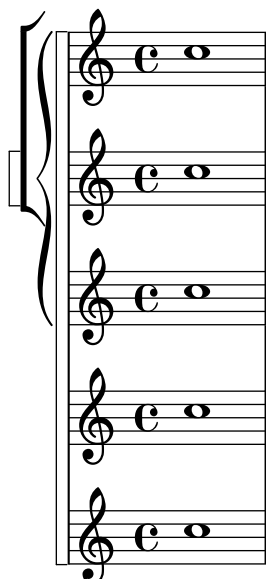
```
\new StaffGroup
\relative c'' <<
  \override StaffGroup.SystemStartSquare.collapse-height = #4
  \set StaffGroup.systemStartDelimiterHierarchy
    = #'(SystemStartSquare (SystemStartBrace (SystemStartBracket a
      (SystemStartSquare b) ) c ) d)

  \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 `,SHARP`, `,FLAT`, `,DOUBLE-SHARP` etc.

Alternatively, using the more concise format for each item in the list, `(step . alter)` specifies the same alteration holds in all octaves. For microtonal scales where a “sharp” is not 100 cents, `alter` refers to the 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 \dwn reb resd
  dod dob dosd \dwn dob |
  dobsb dods do do |
}

```



Numbering groups of measures

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.

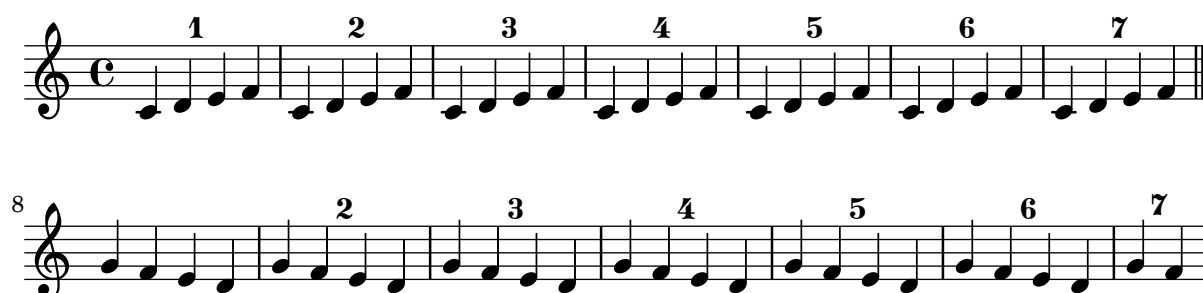
The engraver must be added to the appropriate context. Here, a `Staff` context is used; another possibility is a `Dynamics` context.

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
}
```





Orchestra choir and piano template

This template demonstrates the use of nested `StaffGroup` and `GrandStaff` contexts to sub-group 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 }

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

altoILyrics = \sopranoLyrics

altoIIILyrics = \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 }

violinIMus = \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 }

\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_altoI"
\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
>>
>>
\layout { }
}

```

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

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. The offset needed is established with 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 with transposition

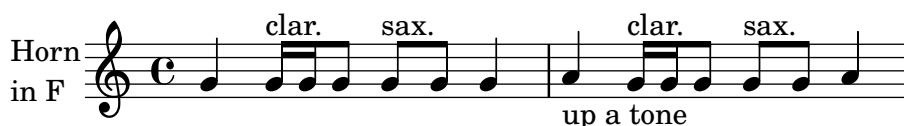
Quotations take into account the transposition of both source and target. In this example, all instruments play sounding middle C; the target is an instrument in F. The target part may be transposed using `\transpose`. In this case, all the pitches (including the quoted ones) are transposed.

```
\addQuote clarinet {
  \transposition 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" >>
}
```



Quoting another voice

The `quotedEventTypes` property determines the music event types which should be quoted. The default value is `(note-event rest-event tie-event beam-event tuplet-span-event)`, which means that only the notes, rests, ties, beams and tuplets of the quoted voice will appear in the `\quoteDuring` expression. In the following example, a 16th rest is not quoted since `rest-event` is not in `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. }
    }
  >>
>>
>>

```

Removing brace on first line of piano score

This snippet removes the first brace from a `PianoStaff` or a `GrandStaff`.

It may be useful when cutting and pasting the engraved image into existing music.

It uses `\alterBroken`.

```

someMusic = {
  \once \override Staff.Clef.stencil = ##f
  \once \override Staff.TimeSignature.stencil = ##f
  \repeat unfold 3 c1 \break
  \repeat unfold 5 c1 \break
}

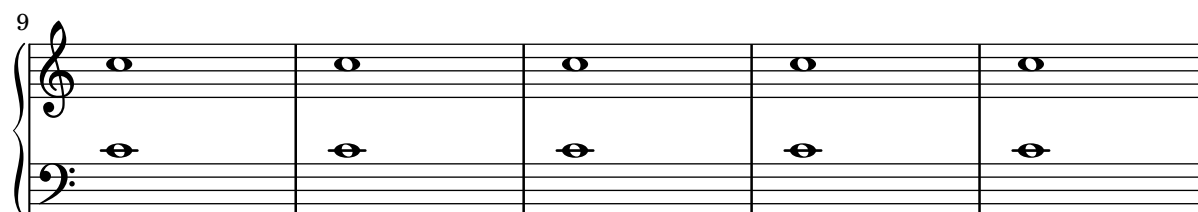
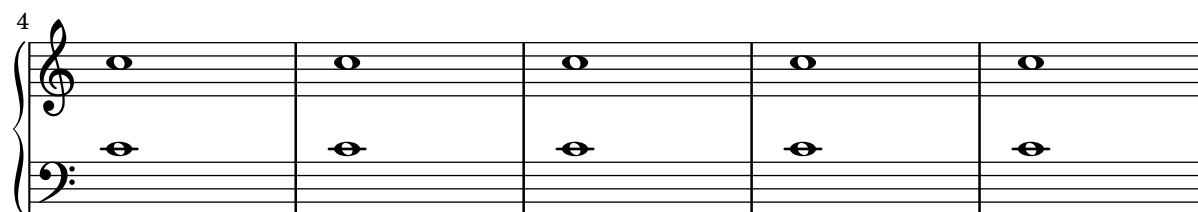
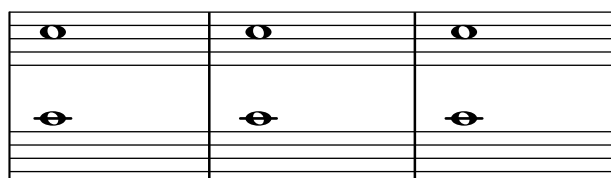
```

```

\repeat unfold 5 c1
}

\score {
  \new PianoStaff
  <<
    \new Staff = "right" \relative c' { \someMusic
    \new Staff = "left" \relative c' { \clef F \someMusic }
  >>
  \layout {
    indent=75
    \context {
      \PianoStaff
      \alterBroken transparent #'(#t) SystemStartBrace
    }
  }
}

```



Removing the first empty line

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

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

```

\layout {
  \context {
    \Staff \RemoveEmptyStaves
    % To use the setting globally, uncomment the following line:
    % \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 image displays musical notation for two systems. The first system consists of a single staff with a treble clef, common time signature, and a sequence of notes: e4, f, g, a, and a whole note c1. The second system consists of two staves. The top staff has a treble clef, common time signature, and a whole note c1. The bottom staff has a treble clef, common time signature, and a whole rest. The two staves are grouped by a brace on the left.

Setting system separators

System separators can be inserted between systems. Any markup can be used, but `\slashSeparator` has been provided as a sensible default.

```
\paper {
```

```
system-separator-markup = \slashSeparator
line-width = 120
}

notes = \relative c' {
  c1 | c \break
  c1 | c \break
  c1 | c
}

\book {
  \score {
    \new GrandStaff <<
      \new Staff \notes
      \new Staff \notes
    >>
  }
}
```

Three systems of musical notation, each consisting of a grand staff (treble and bass clef) and a common time signature 'C'. Each system contains two measures of music. The first system has a whole note in the treble clef and a whole note in the bass clef. The second system has a whole note in the treble clef and a whole note in the bass clef. The third system has a whole note in the treble clef and a whole note in the bass clef. Each system is separated by a double bar line and a repeat sign.

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.defaultBarType = #"'"
  c4 d e f
  g4 f e d
  c4 d e f
  g4 f e d
  \bar "|."
}
```



Time signature in parentheses - method 3

Another way to put the time signature in parenthesis

```
\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 in parentheses

The time signature can be enclosed within parentheses.

```
\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
}
```



Tweaking clef properties

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

For example, `\clef "treble_8"` is equivalent to setting the `clefGlyph`, `clefPosition` (the vertical position of the clef itself on the staff), `middleCPosition` and `clefTransposition`. Note

that when any of these properties (except `middleCPosition`) are changed a new clef symbol is printed.

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

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

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

\set Staff.clefGlyph = #"clefs.F"
\set Staff.clefPosition = #2
c'1
\set Staff.clefGlyph = #"clefs.G"
c'1
\set Staff.clefGlyph = #"clefs.C"
c'1
\set Staff.clefTransposition = #7
```

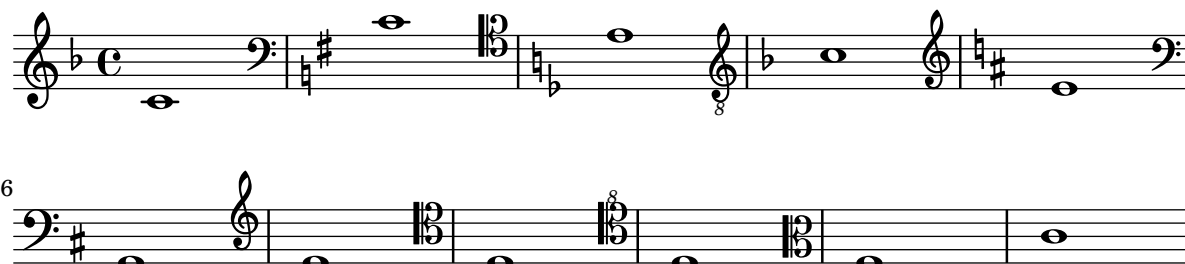
```

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

% Return to the normal clef:

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

```



Two \partCombine pairs on one staff

The `\partCombine` function takes two music expressions each containing a part, and distributes them among four Voices named “two” “one” “solo” and “chords” depending on when and how the parts merged into a common voice. The voices output from `\partCombine` can have their layout properties adjusted in the usual way. Here we define extensions of `\partCombine` to make it easier to put four voices on a staff.

```

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'
  \partCombineUp \soprano \alto
  \partCombineDown \tenor \bass
>>

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

```

```

}

customPartCombineUp =
#(define-music-function (partOne partTwo)
  (ly:music? ly:music?)
  "Take the music in @var{partOne} and @var{partTwo} and return
  a @code{VoiceBox} named @q{Up} containing @code{Voice}s
  that contain @var{partOne} and @var{partTwo} merged into one
  voice where feasible. This variant sets the default voicing
  in the output to use upward stems."
  #{
    \new VoiceBox = "Up" <<
      \context Voice = "one" { \voiceOne }
      \context Voice = "two" { \voiceThree }
      \context Voice = "shared" { \voiceOne }
      \context Voice = "solo" { \voiceOne }
      \context NullVoice = "null" {}
      \partCombine #partOne #partTwo
    >>
  #})

customPartCombineDown = #
(define-music-function (partOne partTwo)
  (ly:music? ly:music?)
  "Take the music in @var{partOne} and @var{partTwo} and return
  a @code{VoiceBox} named @q{Down} containing @code{Voice}s
  that contain @var{partOne} and @var{partTwo} merged into one
  voice where feasible. This variant sets the default voicing
  in the output to use downward stems."
  #{
    \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 #partOne #partTwo
    >>
  #})

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 }

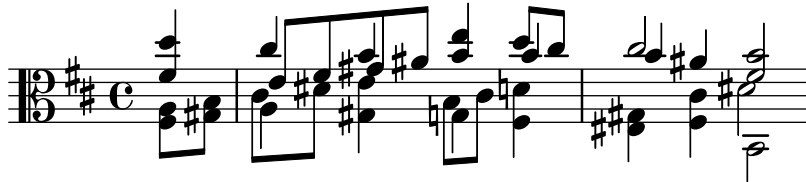
\new Staff <<
  \key b\minor
  \clef alto
  \partial 4
  \transpose b b'

```

```

\customPartCombineUp \soprano \alto
\customPartCombineDown \tenor \bass
>>

```



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

Using `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
  }
}

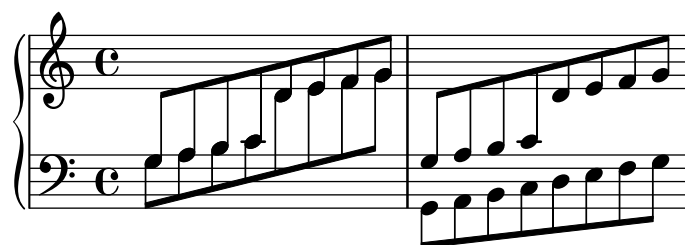
```

```

\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 marklines in a Frenched score

Using `MarkLine` contexts (such as in LSR1010 (<http://lsr.di.unimi.it/LSR/Item?id=1010>)) 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
}

\layout {
  indent = 15\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
  \override VerticalAxisGroup.remove-empty = ##t
  \override VerticalAxisGroup.remove-layer = #'any
  \override VerticalAxisGroup.staff-affinity = #DOWN
  \override VerticalAxisGroup.nonstaff-relatedstaff-spacing.basic-distance = 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
}
}

\score {
  <<
    \new StaffGroup = "winds" \with {
      instrumentName = "Winds"
      shortInstrumentName = "Winds"
    } <<
      \new MarkLine \bars
      \new Staff \winds
    >>
    \new StaffGroup = "brass" <<
      \new MarkLine \bars
      \new Staff = "trumpet" \with {
        instrumentName = "Trumpet"
        shortInstrumentName = "Tpt"
      } \trumpet
      \new Staff = "trombone" \with {
        instrumentName = "Trombone"
        shortInstrumentName = "Tbn"
      } \trombone
    >>
    \new StaffGroup = "strings" \with {

```

```

    instrumentName = "Strings"
    shortInstrumentName = "Strings"
  } <<
    \new MarkLine \bars
    \new Staff = "strings" { \strings }
  >>
>>
}

```

The musical score is divided into four systems, each with a key signature of one flat (B-flat) and a common time signature (C). The tempo and meter are indicated at the beginning of each system.

- System 1:** Tempo: **Allegro** (♩ = 120). Instruments: Winds, Trumpet, Trombone, Strings. Sections: A, B.
- System 2:** Tempo: **Allegro** (♩ = 120). Instruments: Winds, Strings. Sections: C, D.
- System 3:** Tempo: **Adagio** (♩ = 40). Instruments: Winds, Tbn, Strings. Sections: E, F.
- System 4:** Tempo: **Adagio** (♩ = 40). Instruments: Winds, Strings. Sections: G, H.

The score uses standard musical notation, including treble and bass staves, clefs, and various note values. The Winds section is represented by a single staff, while the Trumpet and Trombone sections are represented by separate staves. The Strings section is represented by a single staff.

The image shows a musical score snippet with two systems. The first system (measures 21-25) consists of three staves: Winds, Tpt, and Strings. The second system (measures 26-30) consists of two staves: Winds and Strings. Rehearsal marks J, K, L, M, and N are placed above the staves.

Vertical aligned StaffGroups without connecting SystemStartBar

This snippet shows how to achieve vertically aligned StaffGroups with a SystemStartBar for each StaffGroup, but without connecting them.

% by Thomas Morley

```

#(set-global-staff-size 18)

```

```

\paper {
  indent = 0
  ragged-right = ##f
  print-all-headers = ##t
}

```

```

\layout {
  \context {
    \Staff
    \consists "Mark_engraver"
    \override RehearsalMark.self-alignment-X = #LEFT
  }
  \context {
    \StaffGroup
    systemStartDelimiterHierarchy =
      #'(SystemStartBrace (SystemStartBracket a b))
  }
  \context {
    \Score
    \override SystemStartBrace.style = #'bar-line
    \omit SystemStartBar
    \override SystemStartBrace.padding = #-0.1
    \override SystemStartBrace.thickness = #1.6
    \remove "Mark_engraver"
    \override StaffGroup.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' {
  \mark \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' {
    \mark \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' {
    \mark \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.

Musical notation for the Teacher's simple setting of the choral. It consists of two staves (treble and bass clef) in common time (C). The melody is simple, with notes for 'Wer', 'nur', 'den', 'lie - ben', 'Gott', 'läßt', and 'wal -'. The lyrics are written below the notes.

Pupil: Here's my version!

Musical notation for the Pupil's version of the choral. It consists of two staves (treble and bass clef) in common time (C). The melody is more complex than the Teacher's, with more notes and rests. The lyrics are written below the notes.

Teacher: Well, you simply copied and transposed a version of J.S.Bach.
Do you know this one?

Musical notation for the Teacher's reference to J.S. Bach's version. It consists of two staves (treble and bass clef) in common time (C). The melody is more complex than the Pupil's, with more notes and rests. The lyrics are written below the notes.

ten und hof - fet auf ihn al - le Zeit

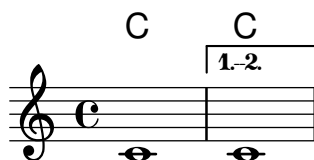
ten und hof - fet auf ihn al - le Zeit

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 under chords.

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

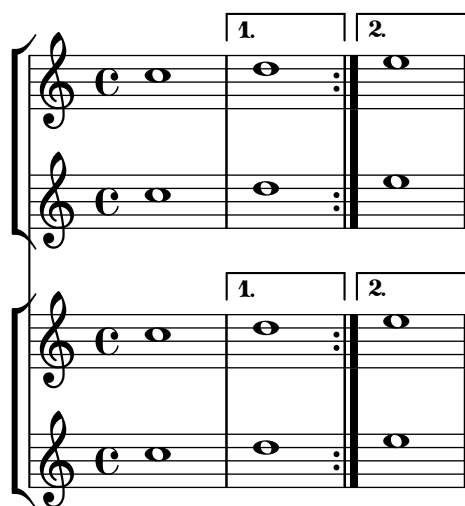


Volta multi staff

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

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

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



Editorial annotations

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 could use `add-link` as defined here. Works with `\override` and `\tweak`.

Drawback: `point-and-click` will be 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)
          (begin
            (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)))
            #f))))
```

```
%%% test
```

```
urlI =
"https://lilypond.org/doc/v2.14/Documentation/notation/writing-pitches"
```

```
urlII =
"https://lilypond.org/doc/v2.14/Documentation/notation/rhythms"
```

```
urlIII =
"https://lilypond.org/doc/v2.14/Documentation/notation/note-heads"
```

```
urlIV =
"https://lilypond.org/doc/v2.14/Documentation/notation/beams"
```

```
urlV =
```

"<https://lilypond.org/doc/v2.14/Documentation/notation/note-head-styles>"

```
\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,
    % With 2.17.9 you could use the command below to address the Accidental.
    % \tweak Accidental.before-line-breaking #(add-link url)
    \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 does not show in a tablature.

To make them appear, simply use the command `\revert TabStaff.TextScript.stencil`

%% <http://lsr.di.unimi.it/LSR/Item?id=919>

% by P.P.Schneider on June 2014

```
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 = #(ly:make-moment 1/8)
    }
  }
}

```

1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 &

T	1-1	1-1	1-1	1-1
A	0-0	0-0	0-0	0-0
B	3	3	2	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

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

```
\relative c'{
  \set Score.alternativeNumberingStyle = #'numbers
  \repeat volta 3 { c4 d e f | }
    \alternative {
      { c4 d e f | c2 d \break }
      { f4 g a b | f4 g a b | f2 a | \break }
      { c4 d e f | c2 d }
    }
  c1 \break
  \set Score.alternativeNumberingStyle = #'numbers-with-letters
  \repeat volta 3 { c,4 d e f | }
    \alternative {
      { c4 d e f | c2 d \break }
      { f4 g a b | f4 g a b | f2 a | \break }
      { c4 d e f | c2 d }
    }
  c1
}
```

The image displays six staves of musical notation, each illustrating a different bar numbering style for repeated music. The notation is in treble clef with a common time signature (C). The first staff, labeled '1', shows a single-measure repeat. The second staff, labeled '2', shows a single-measure repeat. The third staff, labeled '2', shows a three-measure repeat. The fourth staff, labeled '5', shows a single-measure repeat. The fifth staff, labeled '6b', shows a single-measure repeat. The sixth staff, labeled '6c', shows a three-measure repeat.

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
}
```

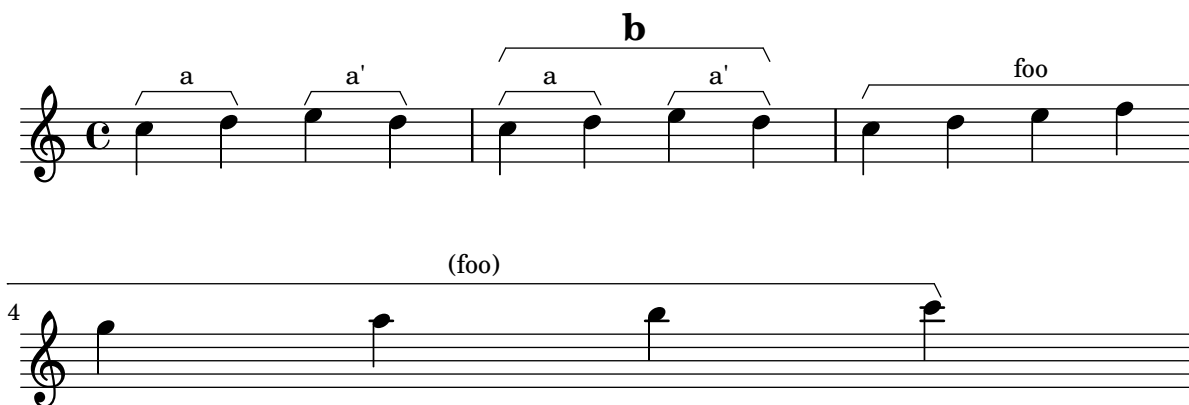


Analysis brackets with labels

Text may be added to analysis brackets through the `text` property of the `HorizontalBracketText` grob. Adding different texts to brackets beginning at the same time requires the `\tweak` command. Bracket text will be 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
}
```

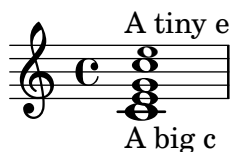


Changing a single note's size in a chord

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)))

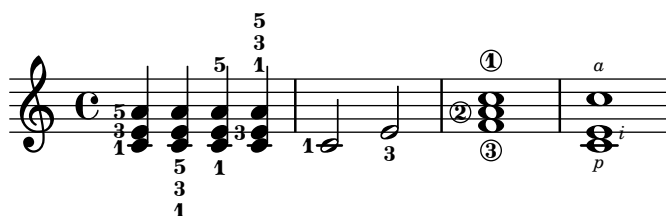
\score {
  \new Staff \relative c' {
    \override NoteHead.color = #color-notehead
    c8 b d dis ees f g aes
  }
}
```



Controlling the placement of chord fingerings

The placement of fingering numbers can be controlled precisely. For fingering orientation to apply, it must be used within a chord construct `<>`, even for single notes. Orientation for string numbers and right-hand fingerings may be set in a similar way.

```
\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 fractions `2/3` and `1/3` adjusts the horizontal position.

```
\relative c' {
  c2*2/3 ( s2*1/3\turn d4) r
  <<
    { c4.( d8) }
    { s4 s\turn }
  >>
  \transpose c d \relative c' <<
    { c4.( d8) }
    {
      s4
      \once \set suggestAccidentals = ##t
      \once \override AccidentalSuggestion.outside-staff-priority = ##f
      \once \override AccidentalSuggestion.avoid-slur = #'inside
    }
  >>
```

```

\once \override AccidentalSuggestion.font-size = -3
\once \override AccidentalSuggestion.script-priority = -1
\single \hideNotes
b8-\turn \noBeam
s8
}
>>
}

```



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 20)

\score {
  {
    \repeat unfold 12 { s1 \break }
  }
  \layout {
    indent = 0\in
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Clef_engraver"
      \remove "Bar_engraver"
    }
    \context {
      \Score
      \remove "Bar_number_engraver"
    }
  }
}

% 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
}
%}

```

```
% uncomment these lines for "A4" size
%{
\paper {
  #(set-paper-size "a4")
  ragged-last-bottom = ##f
  line-width = 180
  left-margin = 15
  bottom-margin = 10
  top-margin = 10
}
%}
```

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?)
  "Sets the @code{font-size} of @code{InstrumentName}."
  The font-size for the initial @code{instrumentName} is taken from the first
  value in @var{font-size-pair}. @code{shortInstrumentName} will get the second
  value of @var{font-size-pair}.
  ")
```

```
;; This code could be changed/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)
                           '()))))
```

```

      (if (pair? siblings)
          (begin
            (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)))
              (cdr siblings))))))
#})

\layout {
  \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.

Vl.



Drawing boxes around grobs

The `print`-function can be overridden to draw a box around an arbitrary grob.

```
\relative c'' {
  \override TextScript.stencil =
    #(make-stencil-boxer 0.1 0.3 ly:text-interface::print)
  c'4^"foo"

  \override Stem.stencil =
    #(make-stencil-boxer 0.05 0.25 ly:stem::print)
  \override Score.RehearsalMark.stencil =
    #(make-stencil-boxer 0.15 0.3 ly:text-interface::print)
  b8

  \revert Stem.stencil
  \revert Flag.stencil
  c4. c4
  \mark "F"
  c1
}
```



Drawing circles around note heads

Here is how to circle a note.

```
circle =
\once \override 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))))

{ \circle c'' }
```

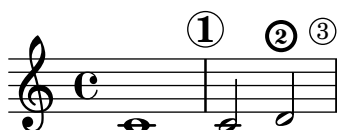


Drawing circles around various objects

The `\circle` markup command draws circles around various objects, for example fingering indications. For other objects, specific tweaks may be required: this example demonstrates two strategies for rehearsal marks and measure numbers.

```
\relative c' {
  c1
  \set Score.markFormatter =
    #(lambda (mark context)
      (make-circle-markup (format-mark-numbers mark context)))
  \mark \default

  c2 d^\markup {
    \override #'(thickness . 3) {
      \circle \finger 2
    }
  }
  \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.

% PostScript is a registered trademark of Adobe Systems Inc.

```
\relative c'' {
  a4-\markup { \postscript "3 4 moveto 5 3 rlineto stroke" }
  -\markup { \postscript "[ 0 1 ] 0 setdash 3 5 moveto 5 -3 rlineto stroke " }

  b4-\markup { \postscript "3 4 moveto 0 0 1 2 8 4 20 3.5 rcurveto stroke" }
  s2
  a'1
}
```



Grid lines: changing their appearance

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

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \relative c'' {
```

```

        \stemUp
        c'4. d8 e8 f g4
    }
}
\new Staff {
    \relative c {
        % this moves them up one staff space from the default position
        \override Score.GridLine.extra-offset = #'(0.0 . 1.0)
        \stemDown
        \clef bass
        \once \override Score.GridLine.thickness = #5.0
        c4
        \once \override Score.GridLine.thickness = #1.0
        g'4
        \once \override Score.GridLine.thickness = #3.0
        f4
        \once \override Score.GridLine.thickness = #5.0
        e4
    }
}
>>
\layout {
    \context {
        \Staff
        % set up grids
        \consists "Grid_point_engraver"
        % set the grid interval to one quarter note
        gridInterval = #(ly:make-moment 1/4)
    }
    \context {
        \Score
        \consists "Grid_line_span_engraver"
        % this moves them to the right half a staff space
        \override NoteColumn.X-offset = #-0.5
    }
}
}

```



Grid lines: emphasizing rhythms and notes synchronization

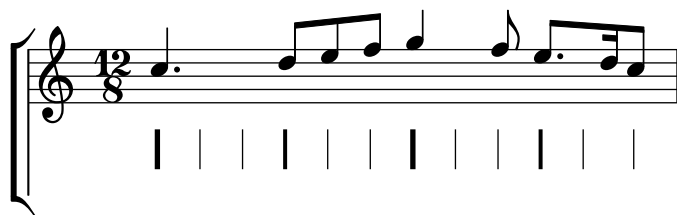
Regular vertical lines can be drawn between staves to show note synchronization; however, in case of monophonic music, you may want to make the second staff invisible, and make the lines shorter like in this snippet.

```

\score {
  \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 = #(ly:make-moment 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)
  }
  ragged-right = ##t
}
}

```



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
  }
}
```



Hammer on and pull off

Hammer-on and pull-off can be obtained using slurs.

```
\new TabStaff {
  \relative c' {
    d4( e\2)
    a( g)
  }
}
```



How to print two rehearsal marks above and below the same barline (method 1)

This method prints two 'rehearsal marks', one on top of the other. It shifts the lower rehearsal mark below the staff and then adds padding above it in order to place the upper rehearsal mark above the staff.

By adjusting the extra-offset and baseline-skip values you can increase or decrease the overall space between the rehearsal mark and the staff.

Because nearly every type of glyph or string can be made to behave like a rehearsal mark it is possible to centre those above and below a bar line.

Adding the appropriate 'break visibility' as shown in snippet 1 (<http://lsr.di.unimi.it/LSR/Item?id=1>) will allow you to position two marks at the end of a line as well.

Note: Method 1 is less complex than Method 2 but does not really allow for fine tuning of placement of one of the rehearsal marks without affecting the other. It may also give some problems with vertical spacing, since using `extra-offset` does not change the bounding box of the mark from its original value.

```
\relative c'{
  c d e f |
  \once \override Score.RehearsalMark.extra-offset = #'(0 . -8.5)
  \once \override Score.RehearsalMark.baseline-skip = #9
  \mark \markup \center-column { \circle 1 \box A }
  g f e d |
  \once \override Score.RehearsalMark.extra-offset = #'(0 . -8.5)
  \once \override Score.RehearsalMark.baseline-skip = #9
  \mark \markup \center-column { \flat { \bold \small \italic Fine. } }
  g f e d |
  \once \override Score.RehearsalMark.extra-offset = #'(0 . -8.5)
  \once \override Score.RehearsalMark.baseline-skip = #9
  \override Score.RehearsalMark.break-visibility = #begin-of-line-invisible
  \mark \markup \center-column { \fermata \box z }
}
```



How to print two rehearsal marks above and below the same barline (method 2)

This method prints two 'rehearsal marks' - one above the stave and one below, by creating two voices, adding the Rehearsal Mark engraver to each voice - without this no rehearsal mark is printed - and then placing each rehearsal mark UP and DOWN in each voice respectively.

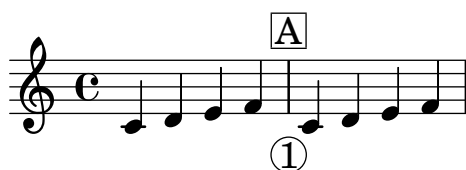
This method (as opposed to method 1) is more complex, but allows for more flexibility, should it be needed to tweak each rehearsal mark independently of the other.

```
\score {
  \relative c'
  <<
  \new Staff {
```

```

<<
  \new Voice \with {
    \consists Mark_engraver
    \consists "Staff_collecting_engraver"
  }
  { c4 d e f
    \mark \markup { \box A }
    c4 d e f
  }
  \new Voice \with {
    \consists Mark_engraver
    \consists "Staff_collecting_engraver"
    \override RehearsalMark.direction = #DOWN
  }
  { s4 s s s
    \mark \markup { \circle 1 }
    s4 s s s
  }
  >>
}
>>
\layout {
  \context {
    \Score
    \remove "Mark_engraver"
    \remove "Staff_collecting_engraver"
  }
}
}

```



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

This example shows how to put crosses on stems. Mark the beginning of a spoken section with the `\speakOn` keyword, and end it with the `\speakOff` keyword.

```
speakOn = {
  \override Stem.stencil =
    #(lambda (grob)
      (let* ((x-parent (ly:grob-parent grob X))
             (is-rest? (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"))
              -2.3))))
}

speakOff = {
  \revert Stem.stencil
  \revert Flag.stencil
}

\score {
  \new Staff {
    \relative c'' {
      a4 b a c
      \speakOn
      g4 f r g
      b4 r d e
      \speakOff
      c4 a g f
    }
  }
}
```



Measure counter

This snippet provides a workaround for emitting measure counters using transparent percent repeats.

```
<<
  \context Voice = "foo" {
    \clef bass
    c4 r g r
    c4 r g r
```

```

    c4 r g r
    c4 r g r
  }
  \context Voice = "foo" {
    \set countPercentRepeats = ##t
    \hide PercentRepeat
    \override PercentRepeatCounter.staff-padding = #1
    \repeat percent 4 { s1 }
  }
>>

```



Numbering groups of measures

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
}

```



Positioning fingering indications precisely

Generally the options available for positioning the fingering of chords work well by default, 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.

```

\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 d-2 a'-5>4
  <c-1 d-\tweak extra-offset #'(0 . 0.2)-2 a'-5>4
  \set fingeringOrientations = #'(down)
  <c-1 d-2 a'-5>4
  <c-\tweak extra-offset #'(0 . -1.1)-1
    d-\tweak extra-offset #'(-1.2 . -1.8)-2 a'-5>4
  \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>4
  \set fingeringOrientations = #'(up)
  <c-1 d-\tweak extra-offset #'(0 . 1.1)-2
    a'-\tweak extra-offset #'(0 . 1)-5>4
  <c-1 d-\tweak extra-offset #'(-1.2 . 1.5)-2
    a'-\tweak extra-offset #'(0 . 1.4)-5>4
}

```



Positioning text markups inside slurs

Text markups need to have the `outside-staff-priority` property set to `false` in order to be printed inside slurs.

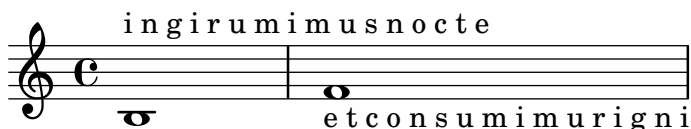
```
\relative c' {
  \override TextScript.avoid-slur = #'inside
  \override TextScript.outside-staff-priority = ##f
  c2(^\markup { \halign #-10 \natural } d4.) c8
}
```



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.

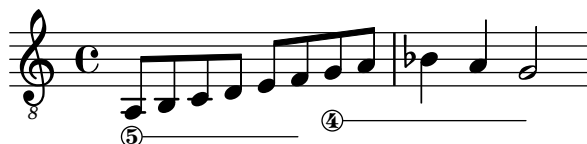
```
stringNumberSpanner =
  #(define-music-function (StringNumber) (string?)
    #{
      \override TextSpanner.style = #'solid
      \override TextSpanner.font-size = #-5
      \override TextSpanner.bound-details.left.stencil-align-dir-y = #CENTER
      \override TextSpanner.bound-details.left.text =
        \markup { \circle \number $StringNumber }
    #})
```

```
\relative c {
  \clef "treble_8"
  \stringNumberSpanner "5"
  \textSpannerDown
  a8\startTextSpan
```

```

b c d e f\stopTextSpan
\stringNumberSpanner "4"
g\startTextSpan a
bes4 a g2\stopTextSpan
}

```



Using PostScript to generate special note head shapes

When a note head with a special shape cannot easily be generated with graphic markup, PostScript code can be used to generate the shape. This example shows how a parallelogram-shaped note head is generated.

```

parallelogram =
  #(ly:make-stencil (list 'embedded-ps
    "gsave
      currentpoint translate
      newpath
      0 0.25 moveto
      1.3125 0.75 lineto
      1.3125 -0.25 lineto
      0 -0.75 lineto
      closepath
      fill
      grestore" )
    (cons 0 1.3125)
    (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
}

```



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 by setting the `whiteout` property of `TimeSignature`. To do this `TimeSignature` is moved to a layer above `Tie`, which is left in the default layer of 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
}
```



Text

Section “Text” in *Notation Reference*

Adding markups in a tablature

By default markups does not show in a tablature.

To make them appear, simply use the command `\revert TabStaff.TextScript.stencil`

```
%% http://lsr.di.unimi.it/LSR/Item?id=919
```

```
% by P.P.Schneider on June 2014
```

```
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 = #(ly:make-moment 1/8)
    }
  }
}
```

	1	&	2	&	3	&	4	&	1	&	2	&	3	&	4	&
T			1	1					1	1						
A			0	0					0	0						
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.

```
% 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!

14-01-2021



14-01-2021

Adjusting lyrics vertical spacing

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 `Staff.InstrumentName #'self-alignment-X` property. 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 }

\score {
  \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 {
  ragged-right = ##t
  indent = 4\cm
  short-indent = 2\cm
}
}

```

Left aligned
instrument name

Centered
instrument name


Right aligned
instrument name



Left

Centered

Right



Aligning objects created with the \mark command

By default the \mark command centers objects over a bar line. This behavior can be modified to align at right or left.

```

\relative c' {
  c1 \mark "(Center)"
  c1
  \once \override Score.RehearsalMark.self-alignment-X = #LEFT
  \mark "(Left)"
  c4 c c c
  c4 c c c
  \once \override Score.RehearsalMark.self-alignment-X = #RIGHT
  \mark "(Right)"
  c1
}

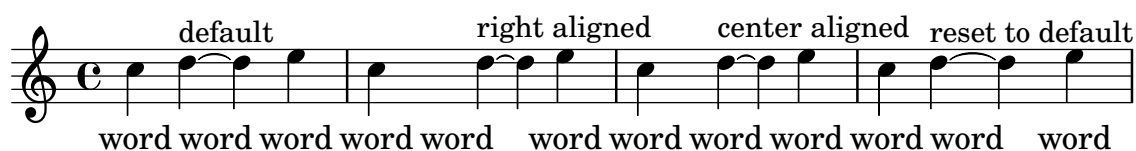
```



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.

```
\score {
  <<
    \new Staff {
      \relative c''
      \new Voice = "vocal" {
        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
    }
  >>
}
```

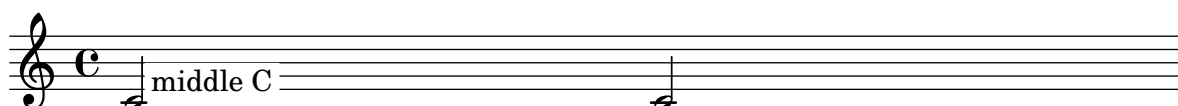


Blanking staff lines using the `\whiteout` command

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 DynamicText.

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))
              (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 text) X CENTER))
                           X LEFT))
              (staff-space (ly:output-def-lookup
                            (ly:grob-layout grob) 'staff-space))
              (staff-line-thickness
               (ly:output-def-lookup (ly:grob-layout grob) 'line-thickness))
              (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 }
```

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

```
\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 with `make-pango-font-tree`.

```
\paper {
  % change for other default global staff size.
  myStaffSize = #20
  %{
    run
      lilypond -dshow-available-fonts
    to show all fonts available in the process log.
  %}

  #(define fonts
    (make-pango-font-tree "Times New Roman"
                          "Nimbus Sans,Nimbus Sans L"
                          "Luxi Mono"
                          "Helvetica"
                          "Courier"
      (/ myStaffSize 20)))
}

\relative c' {
  c4^\markup {
    roman: foo \bold bla \italic bar \italic \bold baz
  }
  c'4_\markup {
    \override #'(font-family . sans)
    {
      sans: foo \bold bla \italic bar \italic \bold baz
    }
  }
  c'2^\markup {
    \override #'(font-family . typewriter)
    {
      mono: foo \bold bla \italic bar \italic \bold baz
    }
  }
}
```



Combining dynamics with markup texts

Some dynamics may involve text indications (such as “più forte” or “piano subito”). These can be produced using a `\markup` block.

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

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

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



Combining two parts on the same staff

The part combiner tool (`\partCombine` command) allows the combination of several different parts on the same staff. Text directions such as “solo” or “a2” are added by default; to remove them, simply set the property `printPartCombineTexts` to `f`. For vocal scores (hymns), there is no need to add “solo/a2” texts, so they should be switched off. However, it might be better not to use it if there are any solos, as they won’t be indicated. In such cases, standard polyphonic notation may be preferable.

This snippet presents the three ways two parts can be printed on a same staff: standard polyphony, `\partCombine` without texts, and `\partCombine` with texts.

%% Combining pedal notes with clef changes

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

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

```
\score {
  <<
    \new Staff \with { instrumentName = "Standard polyphony" }

    << \musicUp \\\musicDown >>

    \new Staff \with {
      instrumentName = "PartCombine without text"
      printPartCombineTexts = ##f
    }

    \partCombine \musicUp \musicDown

    \new Staff \with { instrumentName = "PartCombine with text" }
    \partCombine \musicUp \musicDown
  >>
}
```

```

\layout {
  indent = 6.0\cm
  \context {
    \Score
    \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 will 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 will be regarded as a dynamic, and therefore will remain 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 simultaneous rehearsal marks

Unlike text scripts, rehearsal marks cannot be stacked at a particular point in a score: only one `RehearsalMark` object is created. Using an invisible measure and bar line, an extra rehearsal mark can be added, giving the appearance of two marks in the same column.

This method may also prove useful for placing rehearsal marks at both the end of one system and the start of the following system.

```
{
  \key a \major
  \set Score.markFormatter = #format-mark-box-letters
  \once \override Score.RehearsalMark.outside-staff-priority = #5000
  \once \override Score.RehearsalMark.self-alignment-X = #LEFT
  \once \override Score.RehearsalMark.break-align-symbols = #'(key-signature)
  \mark \markup { \bold { Senza denti } }

  % the hidden measure and bar line
  % \cadenzaOn turns off automatic calculation of bar numbers
  \cadenzaOn
  \once \override Score.TimeSignature.stencil = ##f
  \time 1/16
  s16 \bar ""
  \cadenzaOff

  \time 4/4
  \once \override Score.RehearsalMark.self-alignment-X = #LEFT
  \mark \markup { \box \bold Intro }
  d'1
  \mark \default
  d'1
}
```



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

\set Staff.middleCPosition = #-13
\override TextSpanner.dash-period = #10
\override TextSpanner.dash-fraction = #0.5
\override TextSpanner.thickness = #10
a4 \startTextSpan
b4 c
a4 \stopTextSpan
}

```



Demonstrating all headers

All header fields with special meanings.

```

\header {
  copyright = "copyright"
  title = "title"
  subtitle = "subtitle"
  composer = "composer"
  arranger = "arranger"
  instrument = "instrument"
  metre = "metre"
  opus = "opus"
  piece = "piece"
  poet = "poet"
  texidoc = "All header fields with special meanings."
  copyright = "public domain"
  enteredby = "jcn"
  source = "urtext"
}

\layout {

```



```

    ragged-right = ##f
}

\score {
  \relative c'' { c1 | c | c | c }
}

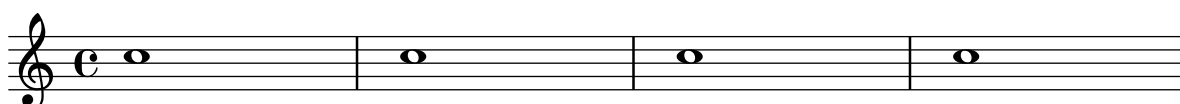
\score {
  \relative c'' { c1 | c | c | c }
  \header {
    title = "localtitle"
    subtitle = "localsubtitle"
    composer = "localcomposer"
    arranger = "localarranger"
    instrument = "localinstrument"
    metre = "localmetre"
    opus = "localopus"
    piece = "localpiece"
    poet = "localpoet"
    copyright = "localcopyright"
  }
}

```

	title	
	subtitle	
poet	instrument	composer
		arranger
piece		opus



localpiece	localopus
------------	-----------



Embedding native PostScript in a \markup block

PostScript code can be directly inserted inside a \markup block.

% PostScript is a registered trademark of Adobe Systems Inc.

```

\relative c'' {
  a4-\markup { \postscript "3 4 moveto 5 3 rlineto stroke" }
  -\markup { \postscript "[ 0 1 ] 0 setdash 3 5 moveto 5 -3 rlineto stroke " }
}

```

```

b4-\markup { \postscript "3 4 moveto 0 0 1 2 8 4 20 3.5 rcurveto stroke" }
s2
a'1
}

```



Formatting lyrics syllables

Markup mode may be used to format individual syllables in lyrics.

```

mel = \relative c'' { c4 c c c }
lyr = \lyricmode {
  Lyrics \markup { \italic can } \markup { \with-color #red contain }
  \markup { \fontsize #8 \bold Markup! }
}

```

```

<<
  \new Voice = melody \mel
  \new Lyrics \lyricsto melody \lyr
>>

```



How to put ties between syllables in lyrics

This can be achieved by separating those syllables by tildes.

```

\lyrics {
  wa~o~a
}

```

waoa

Lyrics alignment

Horizontal alignment for lyrics can be set by overriding the **self-alignment-X** property of the **LyricText** object. #-1 is left, #0 is center and #1 is right; however, you can use **#LEFT**, **#CENTER** and **#RIGHT** as well.

```

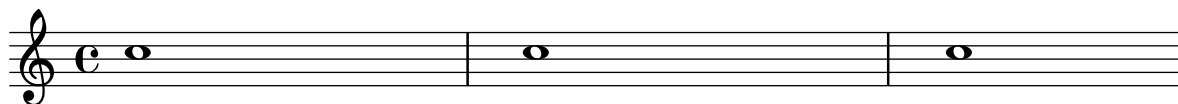
\layout { ragged-right = ##f }
\relative c'' {
  c1
  c1
  c1
}
\addlyrics {
  \once \override LyricText.self-alignment-X = #LEFT
  "This is left-aligned"
}

```

```

\once \override LyricText.self-alignment-X = #CENTER
"This is centered"
\once \override LyricText.self-alignment-X = #1
"This is right-aligned"
}

```



This is left-aligned This is centered This is right-aligned

Markup list

Text that can spread over pages is entered with the `\markuplist` command.

%% updated/modified by P.P.Schneider on Feb. 2014

```

#(set-default-paper-size "a6")

#(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))))

% 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.

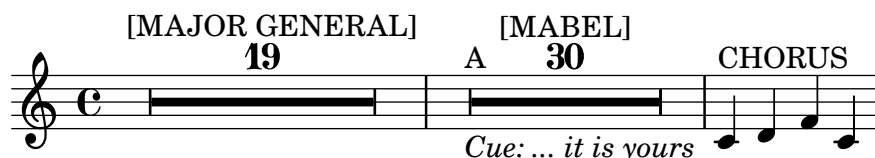
Monsieur le baron était un des plus puissants seigneurs de la Westphalie, car son château avait une porte et des fenêtres. Sa grande salle même était ornée d'une tapisserie. Tous les chiens de ses basses-cours composaient une meute dans le besoin ; ses palefreniers étaient ses piqueurs ; le vicaire du 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 `RehearsalMark` introduced with the `\mark` keyword, 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.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 \text "DynamicText" #})
\score {
  <<
    \new ChordNames
    \with { majorSevenSymbol = \markup "majorSevenSymbol" }
    \chordmode { c1:maj7 }
    \new Staff {
      \tempo \markup "MetronomeMark"
      \mark \markup "RehearsalMark"
      \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

By putting the output of `lilypond-version` into a lyric, it is possible to print the version number of LilyPond in a score, or in a document generated with `lilypond-book`. Another possibility is to append the version number to the doc-string, in this manner:

```

\score {
  \new Lyrics {
    \override Score.RehearsalMark.self-alignment-X = #LEFT
    \mark #(string-append "Processed with LilyPond version " (lilypond-version))
  }
}

```

```
}
}
```

Processed with LilyPond version 2.22.0

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

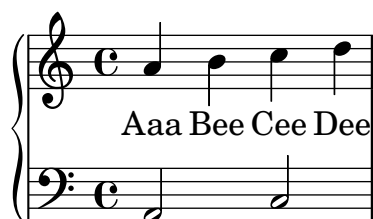
  a4 b c d
}

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 bar numbers with changing regular intervals

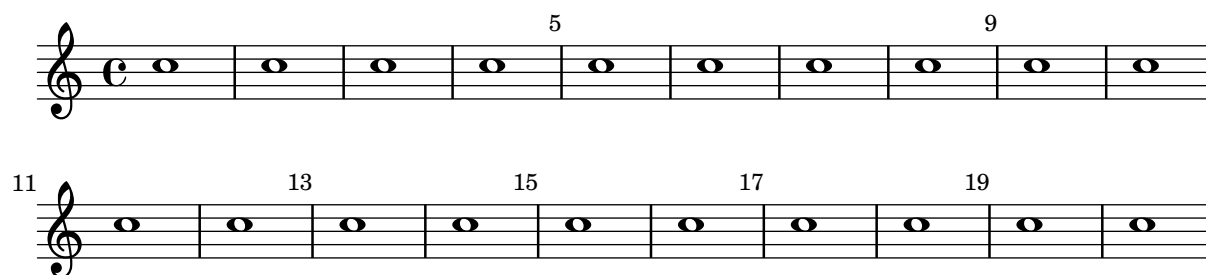
The bar number interval can be changed by changing the context function `{set-bar-number-visibility}`.

```
\relative c' {
```

```

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

```



Printing marks at the end of a line

Marks can be printed at the end of the current line, instead of the beginning of the following line. In such cases, it might be preferable to align the right end of the mark with the bar line.

```

\relative c'' {
  g2 c
  d,2 a'
  \once \override Score.RehearsalMark.break-visibility =
    #end-of-line-visible
  \once \override Score.RehearsalMark.self-alignment-X =
    #RIGHT
  \mark "D.C. al Fine"
  \break
  g2 b,
  c1 \bar "||"
}

```



Printing marks on every staff

Although text marks are normally only printed above the topmost staff, they may also be printed on every staff.

```

\score {
  <<
    \new Staff { c''1 \mark "molto" c'' }
    \new Staff { c'1 \mark "molto" c' }
  >>
  \layout {

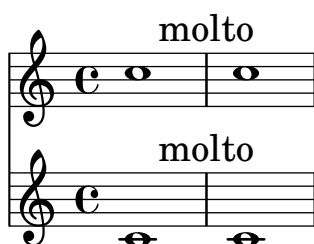
```



```

\context {
  \Score
  \remove "Mark_engraver"
  \remove "Staff_collecting_engraver"
}
\context {
  \Staff
  \consists "Mark_engraver"
  \consists "Staff_collecting_engraver"
}
}
}

```



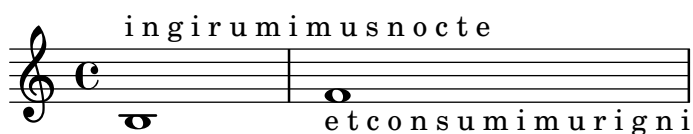
Printing text from right to left

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. The offset needed is established with 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 { O sacrum convivium }
      \line { in quo Christus sumitur, }
      \line { recolitur memoria passionis ejus, }
      \line { mens impletur gratia, }
      \line { futurae gloriae nobis pignus datur. }
      \line { Amen. }
    }
    \hspace #2
    \column \italic {
      \line { O sacred feast }
      \line { in which Christ is received, }
      \line { the memory of His Passion is renewed, }
      \line { the mind is filled with grace, }
      \line { and a pledge of future glory is given to us. }
      \line { Amen. }
    }
  }
  \hspace #1
}
```

O sacrum convivium
in quo Christus sumitur,
recolitur memoria passionis ejus,
mens impletur gratia,
futurae gloriae nobis pignus datur.
Amen.

*O sacred feast
in which Christ is received,
the memory of His Passion is renewed,
the mind is filled with grace,
and a pledge of future glory is given to us.
Amen.*

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.

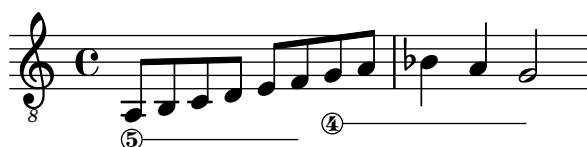
```
stringNumberSpanner =
  \define-music-function (StringNumber) (string?)
  #{
    \override TextSpanner.style = #'solid
```

```

\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"
  \stringNumberSpanner "5"
  \textSpannerDown
  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 @var{arg}. Looks at @code{thickness},
@code{box-padding} and @code{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)))

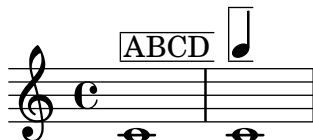
```

```

(NWS-box-stencil m thickness pad)))

% Test it:

\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.

```

% end verbatim - this comment is a hack to prevent texinfo.tex
% from choking on non-European UTF-8 subsets
```

```

%% Edit this file using a Unicode aware editor, such as GVIM, GEDIT, Emacs
```

```

%{
```

You may have to install additional fonts.

Red Hat Fedora

```

linux-libertine-fonts (Latin, Cyrillic, Hebrew)
ipa-mincho-fonts ipa-gothic-fonts (Japanese)
```

Debian GNU/Linux, Ubuntu

```

fonts-linuxlibertine (Latin, Cyrillic, Hebrew)
fonts-ipafont (Japanese)
```

```

%}
```

```

% Font settings for Cyrillic and Hebrew
% Linux Libertine fonts contain Cyrillic and Hebrew glyphs.
```

```

\paper {
  #(define fonts
    (set-global-fonts
      #:roman "Linux Libertine O,serif"
      #:sans "Linux Biolinum O,sans-serif"
      #:typewriter "Linux Libertine Mono O,monospace"
    ))
}
```

```

% Cyrillic font
```

```

bulgarian = \lyricmode {
  Жълтата дюля беше щастлива, че пухът, който цъфна, замръзна като гьон.
}

hebrew = \lyricmode {
  .
}

japanese = \lyricmode {

}

% "a legal song to you"
portuguese = \lyricmode {
  à vo -- cê uma can -- ção legal
}

\relative c' {
  c2 d
  e2 f
  g2 f
  e1
}
\addlyrics { \bulgarian }
\addlyrics { \hebrew }
\addlyrics { \japanese }
\addlyrics { \portuguese }

```

The musical score is written on two staves. The first staff contains four measures of music, and the second staff contains three measures. The lyrics are written in four languages: Bulgarian, Hebrew, Japanese, and Portuguese. The lyrics are aligned both below and above the notes on the staff.

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	אֵיךְ	うめのおくや	can -
6	пухът,	תִּנְצָח	まけふこえて	- ção
7	който	קִרְפָּד	あさきゆめみじ	legal

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 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 { 1. 2. 3... \text \italic { ad lib. } }
```

```

\relative c'' {
  c1
  \set Score.repeatCommands = #(list (list 'volta voltaAdLib) 'start-repeat)
  c4 b d e
  \set Score.repeatCommands = #'((volta #f) (volta "4.") end-repeat)
  f1
  \set Score.repeatCommands = #'((volta #f))
}

```



Vocal music

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 `\splitStaffBarLine`, `convUpStaffBarLine` and `convDownStaffBarLine` commands. 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.

```
#(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 (lambda ()
          (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 {
  ragged-right = ##t
  short-indent = 10\mm
}

```

```

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
}

\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
    }
  }
}

```

S A

T B

a b c d

a b c d

SI SII

AI AII

T B

e f g h

e f g h

e f g h

S A

T B

i j k l

i j k l

Adding orchestral cues to a vocal score

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

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

```

    }
    #})

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

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

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

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

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

```



Adjusting lyrics vertical spacing

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 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.

```
\score {
  <<
    \new Staff {
      \relative c''
```

```

\new Voice = "vocal" {
  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
}
>>
}

```



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 Section “Graphical Object Interfaces” in *Internals Reference* 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
  }
>>

```



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'
    }
  }
}
>>

```



Ancient notation template – modern transcription of gregorian music

This example demonstrates how to do modern transcription of Gregorian music. Gregorian music has no measure, no stems; it uses only half and quarter note heads, and special marks, indicating rests of different length.

```
\include "gregorian.ly"

chant = \relative c' {
  \set Score.timing = ##f
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g) a2 \finalis
}

verba = \lyricmode {
  Lo -- rem ip -- sum do -- lor sit a -- met
}

\score {
  \new Staff <<
    \new Voice = "melody" \chant
    \new Lyrics = "one" \lyricsto melody \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Bar_engraver"
      \hide Stem
    }
    \context {
      \Voice
      \override Stem.length = #0
    }
    \context {
      \Score
      barAlways = ##t
    }
  }
}
```



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 = #(ly:make-moment 1/2)
  }
  \context {
    \Staff
    \remove "Time_signature_engraver"
  }
}
} % End score
} % End markup

\markup {
  \fill-line {
    \column {
      \left-align {
        \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 joice }
          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.

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 this can be done with `\override VerticalAxisGroup.nonstaff-nonstaff-spacing.minimum-distance = ##f`.

```

\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 a grand staff with two staves. The first system shows Alice singing "may-be" and Eve singing "that the". The second system shows Alice singing "sec-ond" and Eve singing "words are". The third system shows Eve singing "...and then I" and Alice singing "Shut up, you like--". The fourth system shows Eve singing "fool!".

Changing stanza fonts

Fonts can be changed independently for each stanza, including the font used for printing the stanza number.

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

```

\time 3/4
g2 e4
a2 f4
g2.
}
\addlyrics {
  \set stanza = #"1. "
  Hi, my name is Bert.
}
\addlyrics {
  \override StanzaNumber.font-name = #"DejaVu Sans"
  \set stanza = #"2. "
  \override LyricText.font-family = #'typewriter
  Oh, ché -- ri, je t'aime
}

```



Chant or psalms notation

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

```

stemOff = \hide Staff.Stem
stemOn  = \undo \stemOff

```

```

\score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \key g \minor
    \cadenzaOn
    \stemOff a'\breve bes'4 g'4
    \stemOn a'2 \bar "||"
    \stemOff a'\breve g'4 a'4
    \stemOn f'2 \bar "||"
    \stemOff a'\breve^{\markup { \italic flexe }}
    \stemOn g'2 \bar "||"
  }
}

```



Forcing hyphens to be shown

If LilyPond does not think there is space for a hyphen, it will be omitted. The 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

Markup mode may be used to format individual syllables in lyrics.

```

mel = \relative c'' { c4 c c c }
lyr = \lyricmode {
  Lyrics \markup { \italic can } \markup { \with-color #red contain }
  \markup { \fontsize #8 \bold Markup! }
}

<<
  \new Voice = melody \mel
  \new Lyrics \lyricsto melody \lyr
>>

```



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 when each line starts and ends with a partial measure. It also shows how to add the verses as stand-alone text under the music.

```
Timeline = {
  \time 4/4
  \tempo 4=96
  \partial 2
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||" \break
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||"
}

SopranoMusic = \relative g' {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

AltoMusic = \relative c' {
  d4 d | d d d d | d d d d | d d d d | d2
  d4 d | d d d d | d d d d | d d d d | d2
}

TenorMusic = \relative a {
  b4 b | b b b b | b b b b | b b b b | b2
  b4 b | b b b b | b b b b | b b b b | b2
}

BassMusic = \relative g {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

global = {
  \key g \major
}

\score { % Start score
  <<
  \new PianoStaff << % Start pianostaff
  \new Staff << % Start Staff = RH
  \global
  \clef "treble"
  \new Voice = "Soprano" << % Start Voice = "Soprano"
  \Timeline
  \voiceOne
  \SopranoMusic
  >> % End Voice = "Soprano"
  \new Voice = "Alto" << % Start Voice = "Alto"
  \Timeline
  \voiceTwo
  \AltoMusic
  >> % End Voice = "Alto"
```



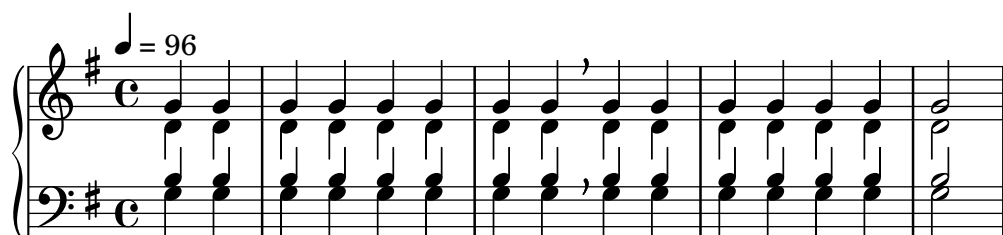
```

>> % End Staff = RH
\new Staff << % Start Staff = LH
  \global
  \clef "bass"
  \new Voice = "Tenor" << % Start Voice = "Tenor"
    \Timeline
    \voiceOne
    \TenorMusic
  >> % End Voice = "Tenor"
  \new Voice = "Bass" << % Start Voice = "Bass"
    \Timeline
    \voiceTwo
    \BassMusic
  >> % End Voice = "Bass"
>> % End Staff = LH
>> % End pianostaff
>>
} % End score

\markup {
  \fill-line {
    ""
    {
      \column {
        \left-align {
          "This is line one of the first verse"
          "This is line two of the same"
          "And here's line three of the first verse"
          "And the last line of the same"
        }
      }
    }
  }
  ""
}

\paper { % Start paper block
  indent = 0 % don't indent first system
  line-width = 130 % shorten line length to suit music
} % End paper block

```



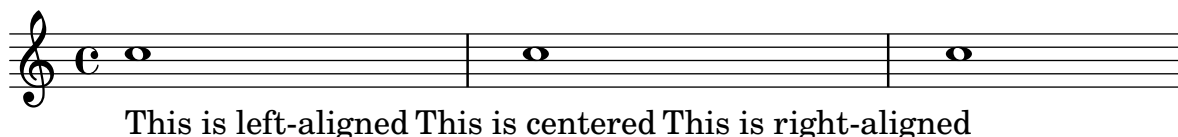


This is line one of the first verse
 This is line two of the same
 And here's line three of the first verse
 And the last line of the same

Lyrics alignment

Horizontal alignment for lyrics can be set by overriding the `self-alignment-X` property of the `LyricText` object. `#-1` is left, `#0` is center and `#1` is right; however, you can use `#LEFT`, `#CENTER` and `#RIGHT` as well.

```
\layout { ragged-right = ##f }
\relative c'' {
  c1
  c1
  c1
}
\addlyrics {
  \once \override LyricText.self-alignment-X = #LEFT
  "This is left-aligned"
  \once \override LyricText.self-alignment-X = #CENTER
  "This is centered"
  \once \override LyricText.self-alignment-X = #1
  "This is right-aligned"
}
```



Marking notes of spoken parts with a cross on the stem

This example shows how to put crosses on stems. Mark the beginning of a spoken section with the `\speakOn` keyword, and end it with the `\speakOff` keyword.

```
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"))
                                -2.3))))
}

speakOff = {
  \revert Stem.stencil
  \revert Flag.stencil
}

\score {
  \new Staff {
    \relative c' {
      a4 b a c
      \speakOn
      g4 f r g
      b4 r d e
      \speakOff
      c4 a g f
    }
  }
}

```



Obtaining 2.12 lyrics spacing in newer versions

The vertical spacing engine changed since version 2.14. This can cause lyrics to be spaced differently.

It is possible to set properties for `Lyric` and `Staff` contexts to get the spacing engine to behave as it did in version 2.12.

```

global = {
  \key d \major
  \time 3/4
}

sopMusic = \relative c' {
  % VERSE ONE
  fis4 fis fis | \break
  fis4. e8 e4
}

altoMusic = \relative c' {
  % VERSE ONE
  d4 d d |
  d4. b8 b4 |
}

tenorMusic = \relative c' {
  a4 a a |
  b4. g8 g4 |
}

```

```

}

bassMusic = \relative c {
  d4 d d |
  g,4. g8 g4 |
}

words = \lyricmode {
  Great is Thy faith -- ful -- ness,
}

\score {
  \new ChoirStaff <<
    \new Lyrics = sopranos
    \new Staff = women <<
      \new Voice = "sopranos" {
        \voiceOne
        \global \sopMusic
      }
      \new Voice = "altos" {
        \voiceTwo
        \global \altoMusic
      }
    >>
    \new Lyrics = "altos"
    \new Lyrics = "tenors"
    \new Staff = men <<
      \clef bass
      \new Voice = "tenors" {
        \voiceOne
        \global \tenorMusic
      }
      \new Voice = "basses" {
        \voiceTwo \global \bassMusic
      }
    >>
    \new Lyrics = basses
    \context Lyrics = sopranos \lyricsto sopranos \words
    \context Lyrics = altos \lyricsto altos \words
    \context Lyrics = tenors \lyricsto tenors \words
    \context Lyrics = basses \lyricsto basses \words
  >>
  \layout {
    \context {
      \Lyrics
      \override VerticalAxisGroup.staff-affinity = ##f
      \override VerticalAxisGroup.staff-staff-spacing =
        #'((basic-distance . 0)
          (minimum-distance . 2)
          (padding . 2))
    }
  }
  \context {

```

```

\Staff
\override VerticalAxisGroup.staff-staff-spacing =
  #'((basic-distance . 0)
    (minimum-distance . 2)
    (padding . 2))
}
}
}

```

The image displays three systems of musical notation for the hymn "Great is Thy faithfulness". Each system consists of three staves: a soprano staff (treble clef), an alto staff (treble clef), and a bass staff (bass clef). The key signature is one sharp (F#) and the time signature is 3/4. The lyrics are: "Great is Thy faithfulness, faithfulness, faithfulness". The first system shows the beginning of the piece. The second system shows the middle of the piece. The third system shows the end of the piece. The lyrics are written below the staves, with hyphens indicating long notes.

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 }
```

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

```
altoILyrics = \sopranoLyrics
```

```
altoIIILyrics = \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 }
```

```
\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_altoI"
  \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" { \altoIIILyrics }
  >>

  \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
>>
>>
\layout { }
}
```


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

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. The offset needed is established with 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

SATB choir template (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
  >>
>>
}

```

p *f*
do do do do
p
re re re re
mp
mi mi mi mi
mf
mi mi mi mi

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 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 { }
}

```

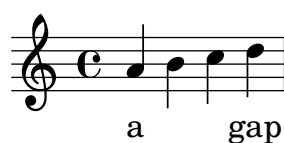


Skips in lyric mode (2)

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.

So for example:

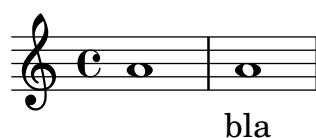
```
<<
\relative c'' { a4 b c d }
\new Lyrics \lyricmode { a4 "" _ gap }
>>
```



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 { \skip 1 bla1 }
>>
```



Using arpeggioBracket to make divisi more visible

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

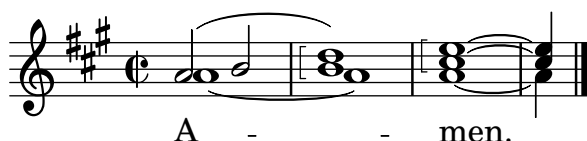
```
\include "english.ly"
```

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

```

\new Voice = "lower"
{ \voiceTwo
  a1 ~
  a
  a ~
  a4 \bar "|."
}
>>
}
\layout { ragged-right = ##t }
}

```



Using tags to produce mensural and modern music from the same source

By using tags, it's possible to use the same music to produce both mensural and modern music. In this snippet, a function `menrest` is introduced, allowing mensural rests to be pitched as in the original, but with modern rests in the standard staff position. Tags are used to produce different types of bar line at the end of the music, but tags can also be used where other differences are needed: for example using “whole measure rests” (`R1`, `R\breve` etc.) in modern music, but normal rests (`r1`, `r\breve`, etc.) in the mensural version. Note that converting mensural music to its modern equivalent is usually referred to as **transcription**.

```

menrest = #(define-music-function (note)
  (ly:music?)
  #{
    \tag #'mens $(make-music 'RestEvent note)
    \tag #'mod $(make-music 'RestEvent note 'pitch '())
  })

MensStyle = {
  \autoBeamOff
  \override NoteHead.style = #'petrucci
  \override Score.BarNumber.transparent = ##t
  \override Stem.neutral-direction = #up
}

finalis = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::finalis
  \once \override BreathingSign.Y-offset = #0
  \once \override BreathingSign.minimum-X-extent = #'(-1.0 . 0.0)
  \once \override BreathingSign.minimum-Y-extent = #'(-2.5 . 2.5)

  \breathe
}

Music = \relative c'' {
  \set Score.tempohideNote = ##t

```



```

\key f \major
\time 4/4
g1 d'2 \menrest bes4 bes2 a2 r4 g4 fis2.
\tag #'mens { \finalis }
\tag #'mod { \bar "||" }
}

MenLyr = \lyricmode { So farre, deere life, deare life }
ModLyr = \lyricmode { So far, dear life, dear life }

\score {
  \keepWithTag #'mens {
    <<
      \new MensuralStaff
      {
        \new MensuralVoice = Cantus
        \clef "mensural-c1" \MensStyle \Music
      }
      \new Lyrics \lyricsto Cantus \MenLyr
    >>
  }
}

\score {
  \keepWithTag #'mod {
    \new ChoirStaff <<
      \new Staff
      {
        \new Voice = Sop \with {
          \remove "Note_heads_engraver"
          \consists "Completion_heads_engraver"
          \remove "Rest_engraver"
          \consists "Completion_rest_engraver" }
        {
          \shiftDurations #1 #0 { \autoBeamOff \Music }
        }
      }
      \new Lyrics \lyricsto Sop \ModLyr
    >>
  }
}

```



Vertically aligning ossias and lyrics

This snippet demonstrates the use of the context properties `alignBelowContext` and `alignAboveContext` to control the positioning of lyrics and ossias.

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

\relative c' <<
  \new Staff = "1" { c4 c s2 }
  \new Staff = "2" { c4 c s2 }
  \new Staff = "3" { c4 c s2 }
  { \skip 2
    <<
      \lyrics {
        \set alignBelowContext = #"1"
        lyrics4 below
      }
      \new Staff \with {
        alignAboveContext = #"3"
        fontSize = #-2
        \override StaffSymbol.staff-space = #(magstep -2)
        \remove "Time_signature_engraver"
      } {
        \tuplet 6/4 {
          \override TextScript.padding = #3
          c8["ossia above" d e d e f]
        }
      }
    >>
  }
>>
```



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. }

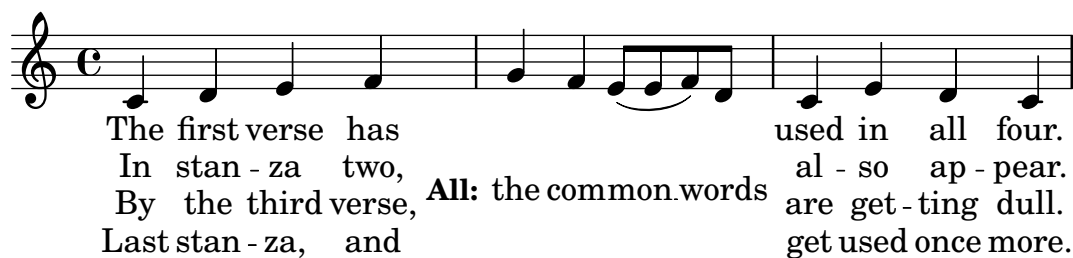
lyricsC = \lyricmode { By the third verse,  \skipFour are get -- ting dull. }

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 used in all four.
 In stan - za two, al - so ap - pear.
 By the third verse, **All:** the common words are get - ting dull.
 Last stan - za, and get used once more.

Vocal ensemble template with automatic piano reduction

This template adds an automatic piano reduction to the standard SATB vocal score demonstrated in “Vocal ensemble template”. This 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 will 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 >>
      >>
    >>
  >>
}

```



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 which 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 |
}

```



```

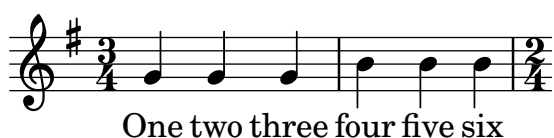
BassLyrics = \lyricmode {
  dum dum |
  dum dum |
}

\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
    }
  }
}

```



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
>>
}

```



Chords

Section “Chord notation” in *Notation Reference*

Adding a figured bass above or below the notes

When writing a figured bass, you can place the figures above or below the bass notes, by defining the `BassFigureAlignmentPositioning.direction` property (exclusively in a `Staff` context). Choices are `#UP` (or `#1`), `#CENTER` (or `#0`) and `#DOWN` (or `#-1`).

This property can be changed as many times as you wish. Use `\once \override` if you don’t want the override to apply to the whole score.

```

bass = {
  \clef bass
  g4 b, c d
  e d8 c d2
}

continuo = \figuremode {
  <_>4 <6>4 <5/>4
  \override Staff.BassFigureAlignmentPositioning.direction = #UP
  %\bassFigureStaffAlignmentUp
  <_+ >4 <6>
  \set Staff.useBassFigureExtenders = ##t
  \override Staff.BassFigureAlignmentPositioning.direction = #DOWN
  %\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 = #'(-2 . 2)
  \consists "Bar_engraver"
}

\chordmode {
  f1:maj7 f:7 bes:7
}

```

F^Δ | F⁷ | B^{b7} |

Bar chords notation for Guitar (with Text Spanner)

Here is how to print bar chords (or barre chords) or half-bar chords (just uncomment the appropriate line for to select either one).

The syntax is : `\bbarre #"fret_number" note(s)`

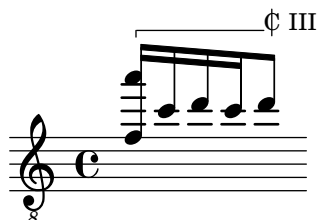
```
%%%%%%%%%% Cut here ----- Start 'bbarred.ly'
```

```
%% C with slash -----
cWithSlash = \markup {
  \combine \roman C \translate #'(0.6 . -0.4) \draw-line #'(0 . 2.0)
}
%% Span -----
%% Syntax: \bbarre #"text" { notes } - text = any number of box
bbarre =
#(define-music-function (barre location str music) (string? ly:music?)
  (let ((elts (extract-named-music music '(NoteEvent EventChord))))
    (if (pair? elts)
        (let ((first-element (first elts))
              (last-element (last elts)))
          (set! (ly:music-property first-element 'articulations)
                (cons (make-music 'TextSpanEvent 'span-direction -1)
                      (ly:music-property first-element 'articulations)))
          (set! (ly:music-property last-element 'articulations)
                (cons (make-music 'TextSpanEvent 'span-direction 1)
                      (ly:music-property last-element 'articulations)))))
        #{}))
  \once \override TextSpanner.font-size = #-2
  \once \override TextSpanner.font-shape = #'upright
  \once \override TextSpanner.staff-padding = #3
  \once \override TextSpanner.style = #'line
  \once \override TextSpanner.to-barline = ##f
  \once \override TextSpanner.bound-details =
    #`((left
      (text . ,#{ \markup { \draw-line #'( 0 . -.5) } #})
      (Y . 0)
      (padding . 0.25)
      (attach-dir . -2))
      (right
      (text . ,#{ \markup { \cWithSlash #str } #})
      (Y . 0)
      (padding . 0.25)
      (attach-dir . 2)))
  %% uncomment this line for make full barred
  % \once \override TextSpanner.bound-details.left.text = \markup{"B" #str}
  $music
  #})

%% %%%%%%%%% Cut here ----- End 'bbarred.ly'
%% Copy and change the last line for full barred. Rename in 'fbarred.ly'
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Syntax: \bbarre #"text" { notes } - text = any number of box
\relative c' { \clef "G_8" \stemUp \bbarre #"III" { <f a'>16[ c' d c d8] } }
```



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 sus4 C^7 | sus4

Changing the chord names to German or semi-German notation

The english naming of chords (default) can be changed to german (`\germanChords` replaces B and Bes with H and B) or semi-german (`\semiGermanChords` replaces B and Bes with H and Bb).

```
scm = \chordmode {
  c1/c | cis/cis
  b1/b | bis/bis | bes/bes
}
```

```
\layout {
  ragged-right = ##t
  \context {
    \ChordNames
    \consists "Instrument_name_engraver"
  }
}
```

```
<<
\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



Changing the positions of figured bass alterations

Accidentals and plus signs can appear before or after the numbers, depending on the `figuredBassAlterationDirection` and `figuredBassPlusDirection` properties.

```

\figures {
  <6\+> <5+> <6 4-> r
  \set figuredBassAlterationDirection = #RIGHT
  <6\+> <5+> <6 4-> r
  \set figuredBassPlusDirection = #RIGHT
  <6\+> <5+> <6 4-> r
  \set figuredBassAlterationDirection = #LEFT
  <6\+> <5+> <6 4-> r
}

```

+6 #5 6 **+6 5# 6** **6+ 5# 6** **6+ #5 6**
 ♭4 4♭ 4♭ ♭4

Chord name exceptions

The property `chordNameExceptions` can be used to store a list of special notations for specific chords.

```

% modify maj9 and 6(add9)
% Exception music is chords with markups

```

```

chExceptionMusic = {
  <c e g b d'>1-\markup { \super "maj9" }
  <c e g a d'>1-\markup { \super "6(add9)" }
}

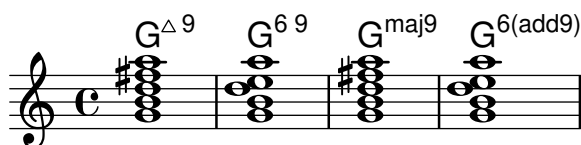
% Convert music to list and prepend to existing exceptions.
chExceptions = #(append
  (sequential-music-to-chord-exceptions chExceptionMusic #t)
  ignatzekExceptions)

theMusic = \chordmode {
  g1:maj9 g1:6.9
  \set chordNameExceptions = #chExceptions
  g1:maj9 g1:6.9
}

\layout {
  ragged-right = ##t
}

<<
  \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^{Δ} 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--2015 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 (_ "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)
  (ly:pitch-transpose pitch
    (ly:make-pitch 0 2 (if (or (= (step-nr pitch) 3)
                              (= (step-nr pitch) 5))
                          FLAT 0))))

(define (step-alteration pitch)
  (let* ((diff (ly:pitch-diff (ly:make-pitch 0 0 0) (car pitches)))
        (normalized-pitch (ly:pitch-transpose pitch diff))
        (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))))
      '()))

```

```

        (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))

;; 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 ^{b3}	C ^{#5}	C ^{b3 b5}
Banter	C _{/no3/no5}	C _{3-/no3/no5}	C _{5+/no3/no5}	C _{3-/5-/no3/no5}
Def	C ⁷	Cm ⁷	C ^Δ	C ^{o7}
Alt	C ⁷	C ^{7 b3}	C ^{#7}	C ^{b3 b5 b7}
Ban	C _{7/no3/no5/no7}	C _{3-/7/no3/no5/no7}	C _{7+/no3/no5/no7}	C _{3-/5-/7-/no3/no5/no7}
Def	C ^{7 #5}	Cm ^Δ	C ^{Δ #5}	C ^o
Alt	C ^{7 #5}	C ^{b3 #7}	C ^{#5 #7}	C ^{7 b3 b5}
Ban	C _{5+/7/no3/no5/no7}	C _{3-/7+/no3/no5/no7}	C _{5+/7+/no3/no5/no7}	C _{3-/5-/7-/no3/no5/no7}
Def	C ⁶	Cm ⁶	C ⁹	Cm ⁹
Alt	C ⁶	C ^{b3 6}	C ⁹	C ^{9 b3}
Ban	C _{6/no3/no5}	C _{3-/6/no3/no5}	C _{9/no3/no5/no7/no9}	C _{3-/9/no3/no5/no7/no9}
Def	Cm ¹³	Cm ¹¹	Cm ^{7 b5 9}	C ^{7 b9}
Alt	C ^{13 b3}	C ^{11 b3}	C ^{9 b3 b5}	C ^{7 b9}
Ban	C _{3-/13/no3/no5/no7/no9/no11+/no13+}	C _{3-/11/no3/no5/no7/no9/no11+}	C _{3-/5-/9/no3/no5/no7/no9}	C _{3-/9/no3/no5/no7/no9}
Def	C ^{7 #9}	C ¹¹	C ^{7 #11}	C ¹³
Alt	C ^{7 #9}	C ¹¹	C ^{9 #11}	C ¹³
Ban	C _{9+/no3/no5/no7/no9}	C _{11/no3/no5/no7/no9/no11+}	C _{11+/no3/no5/no7/no9/no11+}	C _{13/no3/no5/no7/no9/no11+/no13+}

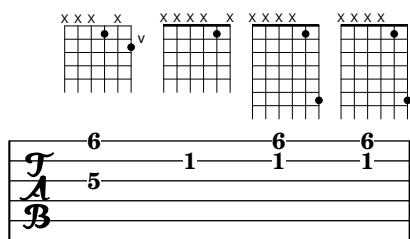
Chords with stretched fingering for FretBoards and TabVoice

Sometimes chords with a stretched fingering are required. If not otherwise specified the context-property `maximumFretStretch` is set to 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.

%% The code below will print two warnings, which may be omitted by uncommenting:
 %#(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
>>

```



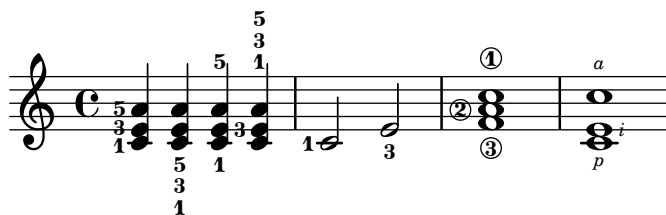
Controlling the placement of chord fingerings

The placement of fingering numbers can be controlled precisely. For fingering orientation to apply, it must be used within a chord construct <>, even for single notes. Orientation for string numbers and right-hand fingerings may be set in a similar way.

```

\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 >
}

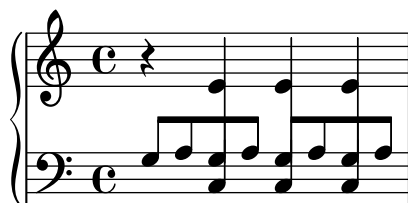
```



Cross-staff chords - beaming problems workaround

Sometimes it is better to use stems from the upper staff for creating cross-staff chords, because no problems with automatic beam collision avoidance then arise. If the stems from the lower staff were used in the following example, it would be necessary to change the automatic beam collision avoidance settings so that it doesn't detect collisions between staves using `\override Staff.Beam.collision-voice-only = ##t`

```
\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 = #11
}
```

```
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
}
```

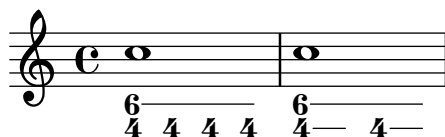


Manually break figured bass extenders for only some numbers

Figured bass often uses extenders to indicate continuation of the corresponding step. However, in this case Lilypond is in greedy-mode and uses extenders whenever possible. To break individual extenders, one can simply use a modifier `\!` to a number, which breaks any extender attributed to that number right before the 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
>>
```



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 and when the chord changes.

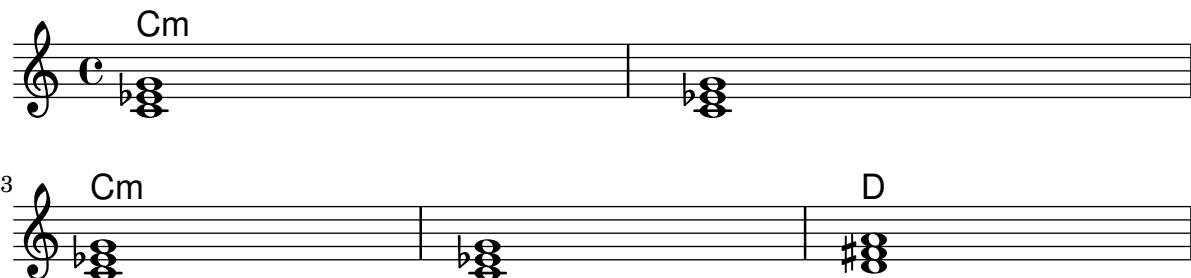
```
harmonies = \chordmode {
  c1:m c:m \break c:m c:m d
}

<<
  \new ChordNames {
    \set chordChanges = ##t
    \harmonies
```

```

}
\new Staff {
  \relative c' { \harmonies }
}
>>

```



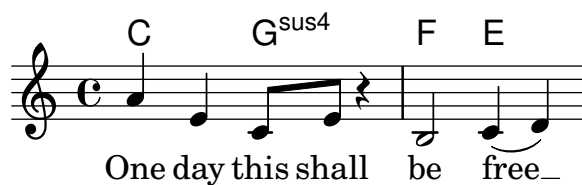
Simple lead sheet

When put together, chord names, a melody, and lyrics form a lead sheet:

```

<<
\chords { c2 g:sus4 f e }
\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
}

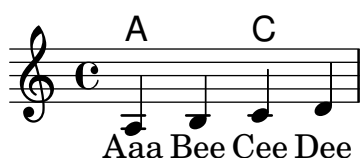
harmonies = \chordmode {
  a2 c
}

```

```

\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 "|"
}

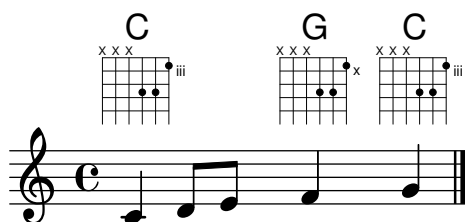
\score {
  <<
    \context ChordNames { \theChords }
    \context FretBoards { \theChords }
    \new Staff {
      \context Voice = "voiceMelody" { \staffMelody }
    }
    \new Lyrics = "lyricsI" {

```

```

\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 true, pairs of congruent figured bass extender lines are vertically centered if `figuredBassCenterContinuations` is set to true.

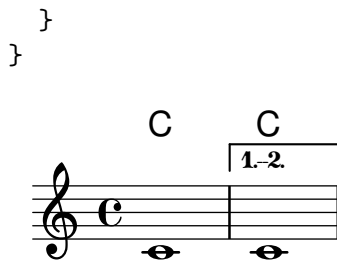
```
<<
\relative c' {
  c8 c b b a a c16 c b b
  c8 c b b a a c16 c b b
  c8 c b b a a c c b b
}
\figures {
  \set useBassFigureExtenders = ##t
  <6+ 4 3>4 <6 4 3>8 r
  <6+ 4 3>4 <6 4 3>8 <4 3+>16 r
  \set figuredBassCenterContinuations = ##t
  <6+ 4 3>4 <6 4 3>8 r
  <6+ 4 3>4 <6 4 3>8 <4 3+>16 r
  \set figuredBassCenterContinuations = ##f
  <6+ 4 3>4 <6 4 3>8 r
  <6+ 4 3>4 <6 4 3>8 <4 3+>8
}
>>
```



Volta below chords

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

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



Keyboards

Section “Keyboard and other multi-staff instruments” in *Notation Reference*

Accordion-discant symbols

This snippet has been obsoleted by predefined markup commands, see ‘Discant symbols’ in the Notation Reference. It’s still useful as a simple demonstration of how to combine symbols: the placement of the symbols added with `\markup` can be tweaked by changing the `\translate-scaled` arguments. `\translate-scaled` is used here rather than `\translate` in order to let the positioning of the symbol parts adapt to changes of `font-size`.

```
discant = \markup {
  \musicglyph "accordion.discant"
}
dot = \markup {
  \musicglyph "accordion.dot"
}

\layout { ragged-right = ##t }

% 16 voets register
accBasson = ^\markup {
  \combine
  \discant
  \translate-scaled #'(0 . 0.5) \dot
}

% een korig 8 en 16 voets register
accBandon = ^\markup {
  \combine
  \discant
  \combine
  \translate-scaled #'(0 . 0.5) \dot
  \translate-scaled #'(0 . 1.5) \dot
}

accVCello = ^\markup {
  \combine
  \discant
  \combine
  \translate-scaled #'(0 . 0.5) \dot
  \combine
  \translate-scaled #'(0 . 1.5) \dot
  \translate-scaled #'(1 . 1.5) \dot
}

% 4-8-16 voets register
accHarmon = ^\markup {
  \combine
  \discant
  \combine
  \translate-scaled #'(0 . 0.5) \dot
```

```

        \combine
        \translate-scaled #'(0 . 1.5) \dot
        \translate-scaled #'(0 . 2.5) \dot
    }

accTrombon = ^\markup {
    \combine
    \discant
    \combine
    \translate-scaled #'(0 . 0.5) \dot
    \combine
    \translate-scaled #'(0 . 1.5) \dot
    \combine
    \translate-scaled #'(1 . 1.5) \dot
    \translate-scaled #'(-1 . 1.5) \dot
}

% eenkorig 4 en 16 voets register
accOrgan = ^\markup {
    \combine
    \discant
    \combine
    \translate-scaled #'(0 . 0.5) \dot
    \translate-scaled #'(0 . 2.5) \dot
}

accMaster = ^\markup {
    \combine
    \discant
    \combine
    \translate-scaled #'(0 . 0.5) \dot
    \combine
    \translate-scaled #'(0 . 1.5) \dot
    \combine
    \translate-scaled #'(1 . 1.5) \dot
    \combine
    \translate-scaled #'(-1 . 1.5) \dot
    \translate-scaled #'(0 . 2.5) \dot
}

accAccord = ^\markup {
    \combine
    \discant
    \combine
    \translate-scaled #'(0 . 1.5) \dot
    \combine
    \translate-scaled #'(1 . 1.5) \dot
    \combine
    \translate-scaled #'(-1 . 1.5) \dot
    \translate-scaled #'(0 . 2.5) \dot
}

```

```
accMusette = ^\markup {
  \combine
  \discant
  \combine
    \translate-scaled #'(0 . 1.5) \dot
  \combine
    \translate-scaled #'(1 . 1.5) \dot
    \translate-scaled #'(-1 . 1.5) \dot
}
```

```
accCeleste = ^\markup {
  \combine
  \discant
  \combine
    \translate-scaled #'(0 . 1.5) \dot
    \translate-scaled #'(-1 . 1.5) \dot
}
```

```
accOboe = ^\markup {
  \combine
  \discant
  \combine
    \translate-scaled #'(0 . 1.5) \dot
    \translate-scaled #'(0 . 2.5) \dot
}
```

```
accClarin = ^\markup {
  \combine
  \discant
  \translate-scaled #'(0 . 1.5) \dot
}
```

```
accPiccolo = ^\markup {
  \combine
  \discant
  \translate-scaled #'(0 . 2.5) \dot
}
```

```
accViolin = ^\markup {
  \combine
  \discant
  \combine
    \translate-scaled #'(0 . 1.5) \dot
  \combine
    \translate-scaled #'(1 . 1.5) \dot
    \translate-scaled #'(0 . 2.5) \dot
}
```

```
\relative c'' {
  c4 d\accBasson e f
  c4 d\accBandon e f
  c4 d\accVCello e f
}
```

```

c4 d\accHarmon e f
c4 d\accTrombon e f
\break
c4 d\accOrgan e f
c4 d\accMaster e f
c4 d\accAccord e f
c4 d\accMusette e f
c4 d\accCeleste e f
\break
c4 d\accOboe e f
c4 d\accClarín e f
c4 d\accPiccolo e f
c4 d\accViolin e f
}

```

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.

```

\use-modules (scm accreg)

```

```

\new PianoStaff
<<
  \new Staff \relative {
    \clef treble
    \discant "10"
    r8 s32 f'[ bes f] s e[ a e] s d[ g d] s16 e32[ a]
    <<
      { r16 <f bes> r <e a> r <d g> }
      \\
      { d r a r bes r }
    >> |
    <cis e a>1
  }

```

```

\new Staff \relative {
  \clef treble
  \freeBass "1"
  r8 d'32 s16. c32 s16. bes32 s16. a32[ cis] s16
  \clef bass \stdBass "Master"
  <<
    { r16 <f, bes d>^"b" r <e a c>^"am" r <d g bes>^"gm" |
      <e a cis>1^"a" }
    \\\
    { d8_"D" c_"C" bes_"B" | a1_"A" }
  >>
}
>>

```

Changing the text for sustain markings

`Staff.pedalSustainStrings` can be used 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 used 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
}

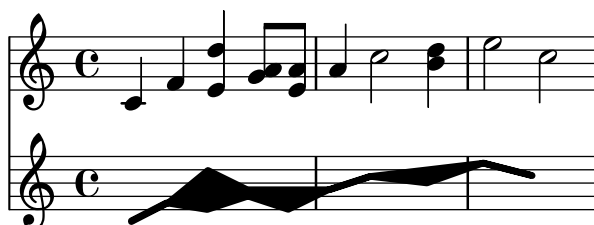
```

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
}

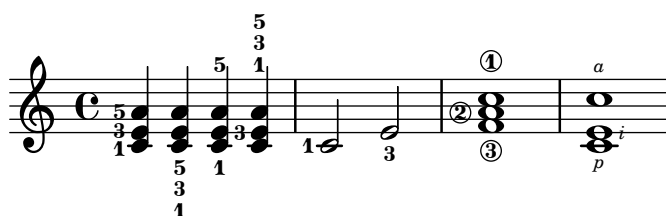
<<
  \new Staff \fragment
  \new Staff \makeClusters \fragment
>>
```



Controlling the placement of chord fingerings

The placement of fingering numbers can be controlled precisely. For fingering orientation to apply, it must be used within a chord construct <>, even for single notes. Orientation for string numbers and right-hand fingerings may be set in a similar way.

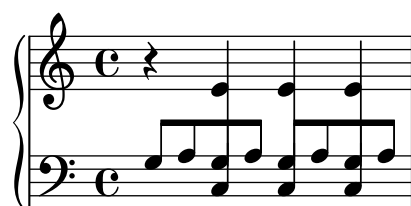
```
\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down right up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(left)
  <c-1>2
  \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 >
}
```




```

        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
  \once \override Staff.PianoPedalBracket.shorten-pair = #'(-7 . -2)
  c2\sostenutoOn c
  c2\sostenutoOff c
  \once \override Staff.PianoPedalBracket.edge-height = #'(0 . 3)
  c2\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.

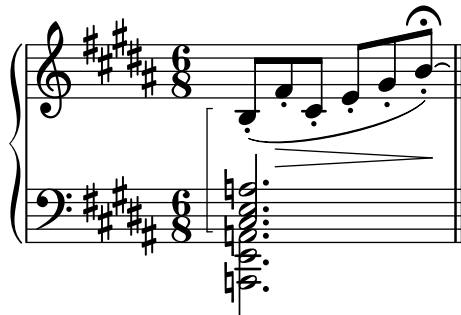
(Debussy, *Les collines d'Anacapri*, m. 65)

```
\new PianoStaff <<
  \set PianoStaff.connectArpeggios = ##t
  \override PianoStaff.Arpeggio.stencil =
    #ly:arpeggio::brew-chord-bracket
  \new Staff {
    \relative c' {
      \key b \major
      \time 6/8
      b8-.(\arpeggio fis'-.\> cis-.
        e-. gis-. b-.)\!\fermata^\laissezVibrer \bar "||"
    }
  }
\new Staff {
  \relative c' {
    \clef bass
    \key b \major
    <<
    {
      <a e cis>2.\arpeggio
    }
  }
}
```

```

      \\\
      {
        <a, e a,>2.
      }
    >>
  }
}
>>

```



Jazz combo template

This is quite an advanced template, for a jazz ensemble. Note that all instruments are notated in `\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 in the documentation
\paper {
  paper-width = 130
}

%#(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
\s1
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
  \s1
  b4 b b b
  \ns1
  c1
}
gtrHarmony = \chordmode {
  \jazzChords
  s1 c2:min7+ d2:maj9
}
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 %%%%%%%%%%%

\score {
  <<
    \new StaffGroup = "horns" <<
      \new Staff = "trumpet" \with { instrumentName = "Trumpet" }
      \trumpet
      \new Staff = "altosax" \with { instrumentName = "Alto Sax" }
      \altoSax
      \new ChordNames = "barichords" \with { instrumentName = "Trumpet" }
      \bariHarmony
      \new Staff = "barisax" \with { instrumentName = "Bari Sax" }
      \bariSax
      \new Staff = "trombone" \with { instrumentName = "Trombone" }
      \trombone
    >>

    \new StaffGroup = "rhythm" <<

```



```

\new ChordNames = "chords" \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

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
```

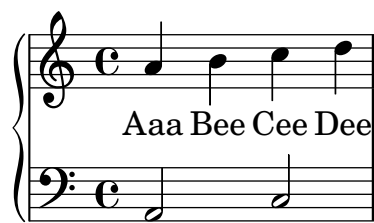
```

    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

```

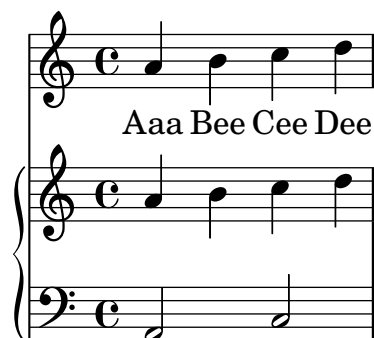
```

\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`.

It may be useful when cutting and pasting the engraved image into existing music.

It uses `\alterBroken`.

```

someMusic = {
  \once \override Staff.Clef.stencil = ##f
  \once \override Staff.TimeSignature.stencil = ##f
  \repeat unfold 3 c1 \break
  \repeat unfold 5 c1 \break
  \repeat unfold 5 c1
}

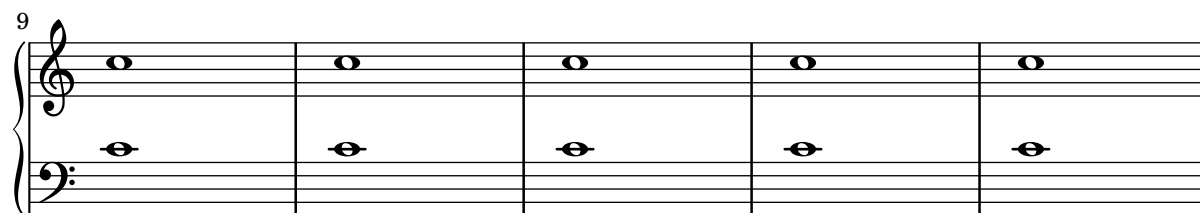
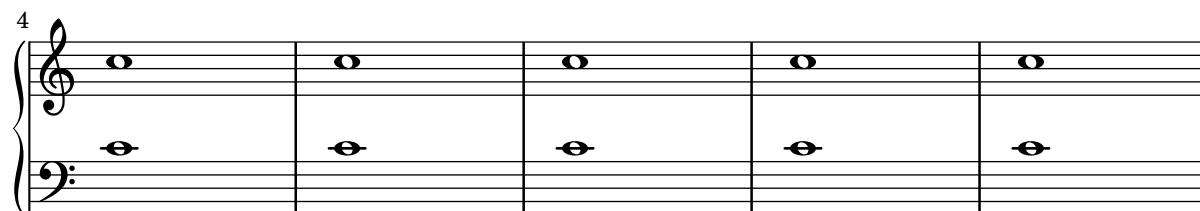
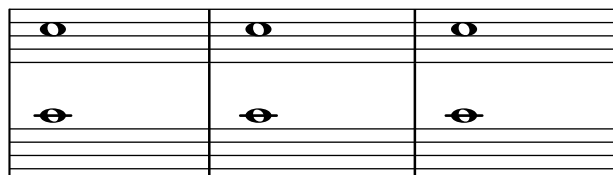
\score {
  \new PianoStaff
  <<
    \new Staff = "right" \relative c'' \someMusic
    \new Staff = "left" \relative c' { \clef F \someMusic }
  >>
  \layout {
    indent=75
  }
}

```

```

\context {
  \PianoStaff
  \alterBroken transparent #'(#t) SystemStartBrace
}
}
}

```



Using autochange with more than one voice

Using 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' {

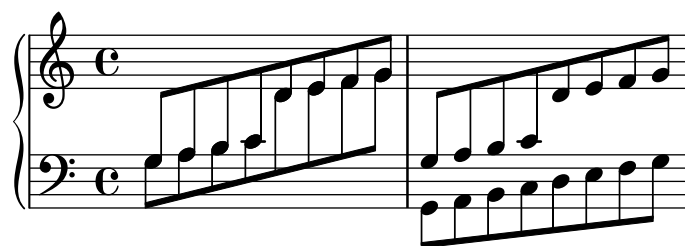
```

```

        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 “Vocal ensemble template”. This 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 will 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 musical score is written for a four-part vocal ensemble (Soprano, Alto, Tenor, Bass) and piano accompaniment. The music is in common time (C) and features a simple, rhythmic melody. The lyrics are 'hi hi hi hi', 'ha ha ha ha', 'hu hu hu hu', and 'ho ho ho ho'.

The vocal parts are arranged as follows:

- Soprano:** The melody is written in the treble clef. The notes are G4, A4, B4, A4, G4, F#4, E4, D4. The lyrics 'hi hi hi hi' are placed above the notes.
- Alto:** The melody is written in the treble clef. The notes are F#4, G4, A4, G4, F#4, E4, D4, C4. The lyrics 'ha ha ha ha' are placed below the notes.
- Tenor:** The melody is written in the bass clef. The notes are E3, F3, G3, F3, E3, D3, C3, B2. The lyrics 'hu hu hu hu' are placed below the notes.
- Bass:** The melody is written in the bass clef. The notes are D3, E3, F3, E3, D3, C3, B2, A2. The lyrics 'ho ho ho ho' are placed below the notes.

The piano accompaniment is written for the right and left hands. The right hand plays a simple harmonic accompaniment, and the left hand plays a simple bass line. The piano part is written in the treble and bass clefs.

Percussion

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 shows two staves of musical notation. The top staff is labeled 'timbales' and the bottom staff is labeled 'drums'. Both staves have a C-clef and a common time signature. The timbales staff contains notes for 'crash' (marked with a cross), 'h.h.' (marked with a plus), and other rhythmic patterns. The drums staff contains notes for various drum parts like 'bd' (bass drum), 'sn' (snare), and 'tom' (tom-tom).

Customized drum notation in printed and MIDI output

Customized drum “pitch” names (suitable for a custom drum style, for example) may be used both in printed and MIDI output by defining such variables as `drumPitchNames`, `drumStyleTable` and `midiDrumPitches`, as demonstrated here. In short, this snippet:

- defines some "pitch" names
- defines how they will be rendered
- tell LilyPond to use them for layout
- assigns pitches to the names
- tell LilyPond to use them for MIDI output

```
%% This snippet tries to amend
```

%% NR 2.5.1 Common notation for percussion - Custom percussion staves

%% <http://lilypond.org/doc/v2.18/Documentation/notation/common-notation-for-percussion#custo>

[illegible]

%%

```
%% To use custom drum-pitch-names for your score and midi you need to follow
```

```
%% this route:
```

%%

0/0/0/0/0/0/0/0/0/0/0/0/0/0/0/0

```
%% LAYOUT:
```

0/0/0/0/0/0/0/0/0/0/0/0/0/0/0

%%

```
%% (1) Define a name and put it in `drumPitchNames'
```

%% This can be done at toplevel with

```
%% drumPitchNames.my-name = #'my-name
```

```
%% It's possible to add an alias as well.
```

```
%% (2) Define how it should be printed
```

```
%% Therefore put them into a toplevel-list, where each entry should look:
```

```
%% (my-name
```

```
%% note-head-style-or-default
```

```
%% articulation-string-or-#f
```

```
%%                                staff-position)
```

%% Example:

```
%%          # (define my-style
```

 $\frac{00}{00}$

```
%% (my-name default "tenuto" -1)
```

%% ; . . .

%%))

```
%% (3) Tell LilyPond to use this custom-definitions, with
```

```
%%      drumStyleTable = #(alist->hash-table my-style)
```

```
%% in a \layout or \with
```

%%

```
%%      Now we're done for layout, here a short, but complete example:
```

```
%% \new DrumStaff
```

```
%% \with { drumStyleTable = #(alist->hash-table my-style) }
```

```
%% \drummode { my-name }
```

%%

0/0/0/0/0/0/0/0/0/0/0/0/0/0/0

```
%% MIDI:
```

%%%%%%%%%

```

%%
%% (1) Again at toplvel, assign a pitch to your custom-note-name
%%      midiDrumPitches.my-name = ges
%%      Note that you have to use the name, which is in drumPitchNames, no alias
%% (2) Tell LilyPond to use this pitch(es), with
%%      drumPitchTable = #(alist->hash-table midiDrumPitches)
%%
%%      Example:
%%      \score {
%%        \new DrumStaff
%%        \with {
%%          drumStyleTable = #(alist->hash-table my-style)
%%          drumPitchTable = #(alist->hash-table midiDrumPitches)
%%        }
%%        \drummode { my-name4 }
%%      \layout {}
%%      \midi {}
%%    }
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% TESTING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% To test whether all is fine, run the following sequence in terminal:
%%      lilypond my-file.ly
%%      midi2ly my-file.midi
%%      gedit my-file-midi.ly
%%
%% Which will do:
%% 1. create pdf and midi
%% 2. transform the midi back to a .ly-file
%%    (note: midi2ly is not always good in correctly identifying enharmonic pitches)
%% 3. open this file in gedit (or use another editor)
%% Now watch what you've got.
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% FULL EXAMPLE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

drumPitchNames.dbass      = #'dbass
drumPitchNames.dba        = #'dbass  % 'db is in use already
drumPitchNames.dbassmute  = #'dbassmute
drumPitchNames.dbm        = #'dbassmute
drumPitchNames.do         = #'dopen
drumPitchNames.dopenmute  = #'dopenmute
drumPitchNames.dom        = #'dopenmute
drumPitchNames.dslap      = #'dslap
drumPitchNames.ds         = #'dslap
drumPitchNames.dslapmute  = #'dslapmute
drumPitchNames.dsm        = #'dslapmute

```

```

#(define djembe
  '((dbass      default  #f          -2)
    (dbassmute  default  "stopped"  -2)
    (dopen      default  #f          0)
    (dopenmute  default  "stopped"  0)
    (dslap      default  #f          2)
    (dslapmute  default  "stopped"  2)))

midiDrumPitches.dbass = g
midiDrumPitches.dbassmute = fis
midiDrumPitches.dopen = a
midiDrumPitches.dopenmute = gis
midiDrumPitches.dslap = b
midiDrumPitches.dslapmute = ais

one = \drummode { r4 dba4 do ds r dbm dom dsm }

\score {
  \new DrumStaff
  \with {
    \override StaffSymbol.line-count = #3
    instrumentName = #"Djembe "
    drumStyleTable = #(alist->hash-table djembe)
    drumPitchTable = #(alist->hash-table midiDrumPitches)
  }
  \one
  \layout {}
  \midi {}
}

```



Heavily customized polymetric time signatures

Though the polymetric time signature shown was 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!).

```

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))
  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
  c4. d4 c4 d4. c4 d c2 d4. e4-~ d4
  c4. d4 c4 d4. c4 d c2 d4. e4-~ d4 \break
}

```

```

drum = \new DrumStaff \drummode {
  \bar ".|:" bd4.^ \markup { Drums } sn4 bd \bar ";" sn4.
}

```

```

bd4 sn \bar ";" bd sn bd4. sn4 bd \bar " :|. "
}

\new Staff \with {
  instrumentName = \markup { \concat { "B" \flat " Sop." } }
}

{
  \melody
  \drum
}

```

B \flat Sop.

Drums

Jazz combo template

This is quite an advanced template, for a jazz ensemble. Note that all instruments are notated in `\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 in the documentation

```

```

\paper {
  paper-width = 130
}
%#(set-global-staff-size 16)
\include "english.ly"

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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
  \nsl
}
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
}
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 %%%%%%%%%%%

\score {
  <<

```

```

\new StaffGroup = "horns" <<
  \new Staff = "trumpet" \with { instrumentName = "Trumpet" }
  \trumpet
  \new Staff = "altosax" \with { instrumentName = "Alto Sax" }
  \altoSax
  \new ChordNames = "barichords" \with { instrumentName = "Trumpet" }
  \bariHarmony
  \new Staff = "barisax" \with { instrumentName = "Bari Sax" }
  \bariSax
  \new Staff = "trombone" \with { instrumentName = "Trombone" }
  \trombone
>>

\new StaffGroup = "rhythm" <<
  \new ChordNames = "chords" \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^{Δ} Solo $C\sharp m^7$

Cm^{Δ} $D^{\Delta 9}$

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 . 5) #'(0 . 5)
  \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 #t  4)
                   (snare      default #t -4)
                   (tambourine default #t  0)))

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

drumsA = {
  \context DrumVoice <<
  { \global }
  { \drummode {
    \autoBeamOff
    \stemDown sn8 \stemUp tamb s8 |
    sn4 \stemDown sn4 |
  }
}

```

```

        \stemUp tamb8 \stemDown sn8 \stemUp sn16 \stemDown sn \stemUp sn8 |
        \stemDown sn8 \stemUp tamb s8 |
        \stemUp sn4 s8 \stemUp tamb
    }
}
>>
}

drumsB = {
  \drummode {
    s4 bd8 s2*2 s4 bd8 s4 bd8 s8
  }
}

\layout {
  indent = #40
}

\score {
  \new StaffGroup <<
    \new DrumStaff \with {
      instrumentName = \markup {
        \center-column {
          "Tambourine"
          "et"
          "caisse claire s. timbre"
        }
      }
    }
    drumStyleTable = #(alist->hash-table mydrums)
  }
  \drumsA
  \new DrumStaff \with {
    instrumentName = #"Grosse Caisse"
    drumStyleTable = #(alist->hash-table mydrums)
  }
  \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.

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

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

\layout {
  \context {
    \Score
    \remove "Timing_translator"
    \remove "Default_bar_line_engraver"
    \remove "Bar_number_engraver"
    \override SpacingSpanner.uniform-stretching = ##t
    \override SpacingSpanner.strict-note-spacing = ##t
    proportionalNotationDuration = #(ly:make-moment 1/64)
  }
  \context {
    \Staff
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  }
  \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 {
  \bar "|"
  \clef treble
  \time 3/8
  d''4.

  \bar "|"
  \time 3/4
  r8 des''2( c''8)

  \bar "|"
```

```

\time 7/8
r4. ees''2 ~

\bar "|"
\time 2/4
\tupletUp
\tuplet 3/2 { ees''4 r4 d''4 ~ }

\bar "|"
\time 3/8
\tupletUp
\tuplet 4/3 { d''4 r4 }

\bar "|"
\time 2/4
e''2

\bar "|"
\time 3/8
es''4.

\bar "|"
\time 3/4
r8 d''2 r8
\bar "|"
}
>>

```

```

Percussion = \new StaffGroup <<
\new Staff <<
\global {
\bar "|"
\clef percussion
\time 3/4
r4 c'2 ~

\bar "|"
c'2.

\bar "|"
R2.

\bar "|"
r2 g'4 ~

\bar "|"
g'2. ~

\bar "|"
g'2.
}
>>

```

```

\new Staff <<
  \global {
    \bar "|"
    \clef percussion
    \time 3/4
    R2.

    \bar "|"
    g'2. ~

    \bar "|"
    g'2.

    \bar "|"
    r4 g'2 ~

    \bar "|"
    g'2 r4

    \bar "|"
    g'2.
  }
>>
>>

\score {
  <<
    \Bassklarinette
    \Percussion
  >>
}

```

The musical score is written for three instruments: Percussion, Bass Clarinet, and Bassoon. The time signature is 3/4. The Percussion part (bottom staff) consists of three measures: the first measure has a quarter note, the second measure has a half note, and the third measure has a quarter note. The Bass Clarinet part (middle staff) consists of three measures: the first measure has a quarter note, the second measure has a half note, and the third measure has a quarter note. The Bassoon part (top staff) consists of three measures: the first measure has a quarter note, the second measure has a half note, and the third measure has a quarter note. The score is written in treble clef for the Bassoon and bass clef for the other two instruments.

(4)

8

Tam-tam example

A tam-tam example, entered with 'tt'

```
#(define mydrums '((tamtam default #t 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

Fretted strings

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
}
```

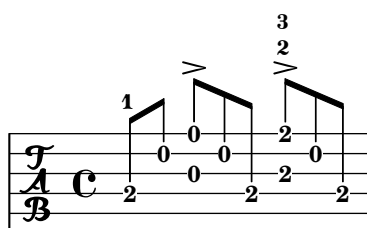


Adding fingerings to tablatures

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
  }
}
threeFour = \markup {
  \override #'(baseline-skip . 2)
  \column {
    \finger 3
    \finger 4
  }
}

\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]
  }
}
```



Adding markups in a tablature

By default markups does not show in a tablature.

To make them appear, simply use the command `\revert TabStaff.TextScript.stencil`

```
%% http://lsr.di.unimi.it/LSR/Item?id=919
```

```
% by P.P.Schneider on June 2014
```

```
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 = #(ly:make-moment 1/8)
    }
  }
}
```

	1	&	2	&	3	&	4	&	1	&	2	&	3	&	4	&
T	1	1							1	1						
A	0	0							0	0						
B	3				3			2	3				3			2
					3								3			

Allowing fingerings to be printed inside the staff

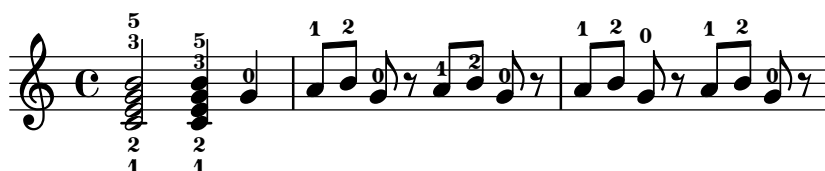
By default, vertically oriented fingerings are positioned outside the staff; that behavior, however, may be disabled. Attention needs to be paid to situations where fingerings and stems are in the same direction: by default, fingerings will avoid only beamed stems. That setting can be changed to avoid 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
}

```



Barres in automatic fretboards

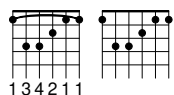
When automatic fretboards are used, barre indicators will be drawn whenever one finger is responsible for multiple strings.

If no finger indications are given in the chord from which the automatic fretboard is created, no barre indicators will be included, because there is no way to identify where barres 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
}

```



Bar chords notation for Guitar (with Text Spanner)

Here is how to print bar chords (or barre chords) or half-bar chords (just uncomment the appropriate line for to select either one).

```

The syntax is : \bbarre #"fret_number" note(s)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%%%%%% Cut here ----- Start 'bbarred.ly'

%% C with slash -----
cWithSlash = \markup {
  \combine \roman C \translate #'(0.6 . -0.4) \draw-line #'(0 . 2.0)
}
%% Span -----
%% Syntax: \bbarre #"text" { notes } - text = any number of box
bbarre =
#(define-music-function (barre location str music) (string? ly:music?)
  (let ((elts (extract-named-music music '(NoteEvent EventChord))))
    (if (pair? elts)
      (let ((first-element (first elts))
            (last-element (last elts)))
        (set! (ly:music-property first-element 'articulations)
              (cons (make-music 'TextSpanEvent 'span-direction -1)
                    (ly:music-property first-element 'articulations))))
      (ly:music-property first-element 'articulations))))

```

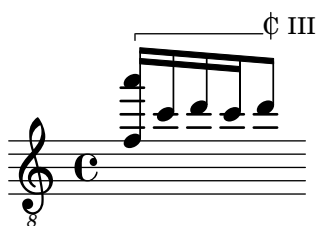
```

      (set! (ly:music-property last-element 'articulations)
        (cons (make-music 'TextSpanEvent 'span-direction 1)
          (ly:music-property last-element 'articulations))))))
#{
  \once \override TextSpanner.font-size = #-2
  \once \override TextSpanner.font-shape = #'upright
  \once \override TextSpanner.staff-padding = #3
  \once \override TextSpanner.style = #'line
  \once \override TextSpanner.to-barline = ##f
  \once \override TextSpanner.bound-details =
    #`((left
      (text . ,#{ \markup { \draw-line #'( 0 . -.5) } #})
      (Y . 0)
      (padding . 0.25)
      (attach-dir . -2))
      (right
      (text . ,#{ \markup { \cWithSlash #str } #})
      (Y . 0)
      (padding . 0.25)
      (attach-dir . 2)))
%% uncomment this line for make full barred
  % \once \override TextSpanner.bound-details.left.text = \markup{"B" #str}
  $music
#})

%% %%%%%%%%% Cut here ----- End 'bbarred.ly'
%% Copy and change the last line for full barred. Rename in 'fbarred.ly'
%% %%%%%%%%%%
%% %%%%%%%%%%

%% Syntax: \bbarre #"text" { notes } - text = any number of box
\relative c' { \clef "G_8" \stemUp \bbarre #"III" { <f a'>16[ c' d c d8] } }

```



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
  }

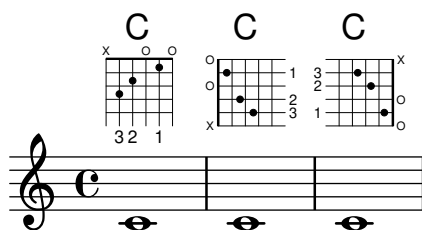
```



```

\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 glissando in tablature

Slides for chords are indicated by default in both **Staff** and **TabStaff**.

String numbers are necessary for **TabStaff** because automatic string calculations are different for chords and for single notes.

```

myMusic = \relative c' {
  <c e g>1 \glissando <f a c>
}

\score {
  <<
    \new Staff {
      \clef "treble_8"
      \myMusic
    }
    \new TabStaff \myMusic
  >>
}

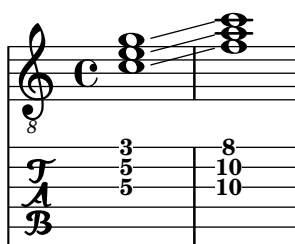
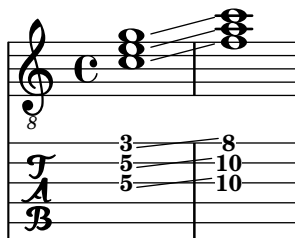
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \myMusic
    }
    \new TabStaff \with { \override Glissando.style = #'none } {

```

```

\myMusic
}
>>
}

```



ChordChanges for FretBoards

FretBoards can be set to display only when the chord changes or at the beginning of a new line.

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

```

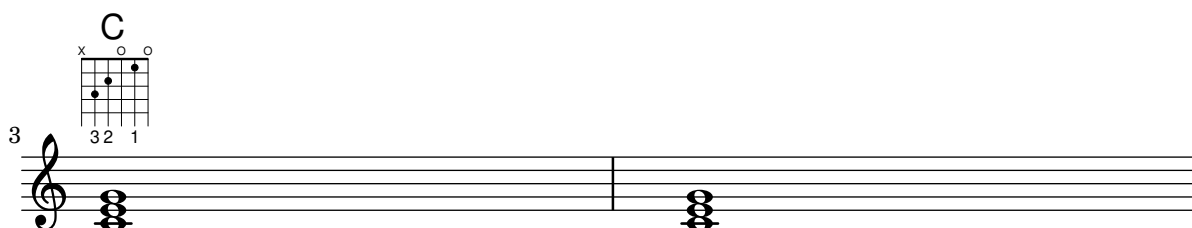
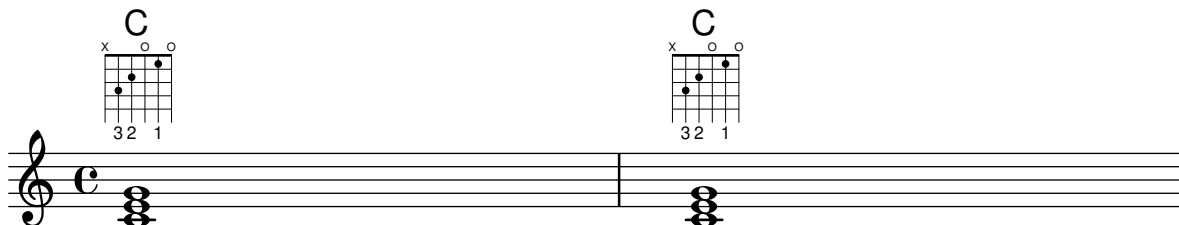
myChords = \chordmode {
  c1 c1 \break
  \set chordChanges = ##t
  c1 c1 \break
  c1 c1
}

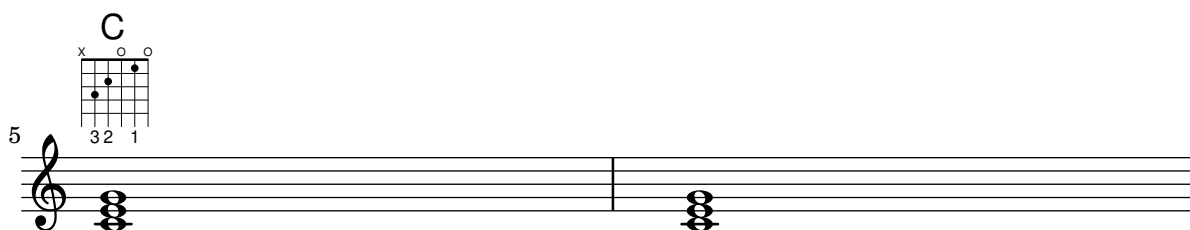
```

```

<<
  \new ChordNames { \myChords }
  \new FretBoards { \myChords }
  \new Staff { \myChords }
>>

```





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 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.

```
% The code below will print two warnings, which may be omitted by uncommenting:
%#(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

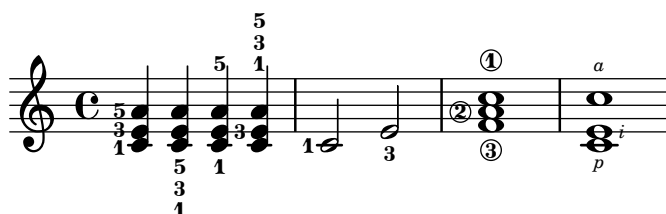
The placement of fingering numbers can be controlled precisely. For fingering orientation to apply, it must be used within a chord construct `<>`, even for single notes. Orientation for string numbers and right-hand fingerings may be set in a similar way.

```
\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 >
}

```



Customizing fretboard fret diagrams

Fret diagram properties can be set through 'fret-diagram-details'. For FretBoard fret diagrams, overrides are applied to the FretBoards.FretBoard object. Like Voice, FretBoards is a bottom level context, therefore can be omitted in property overrides.

```

\include "predefined-guitar-fretboards.ly"
\storePredefinedDiagram #default-fret-table \chordmode { c' }
    #guitar-tuning
    #"x;1-1-(;3-2;3-3;3-4;1-1-);"

% shorthand
oo = #(define-music-function
  (grob-path value)
  (list? scheme?)
  #{ \once \override $grob-path = #value #})

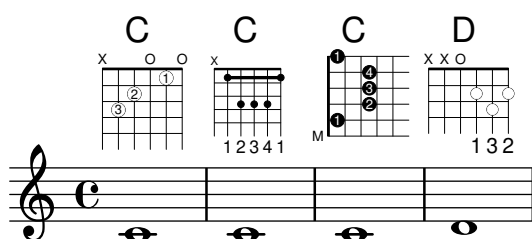
<<
\new ChordNames {
  \chordmode { c1 | c | c | d }
}
\new FretBoards {
  % Set global properties of fret diagram
  \override FretBoards.FretBoard.size = #'1.2
  \override FretBoard.fret-diagram-details.finger-code = #'in-dot
  \override FretBoard.fret-diagram-details.dot-color = #'white
  \chordmode {
    c
    \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
    }
  }
  \new Voice {
    c'1 | c' | c' | d'
  }
>>

```



Customizing markup fret diagrams

Fret diagram properties can be set through 'fret-diagram-details. For markup fret diagrams, overrides can be applied to the Voice.TextScript object or directly to the markup.

```

<<
  \chords { c1 | c | c | d }

  \new Voice = "mel" {
    \textLengthOn
    % Set global properties of fret diagram
    \override TextScript.size = #'1.2
    \override TextScript.fret-diagram-details.finger-code = #'in-dot
    \override TextScript.fret-diagram-details.dot-color = #'white

    %% C major for guitar, no barre, using defaults
    % terse style
    c'1~\markup { \fret-diagram-terse "x;3-3;2-2;o;1-1;o;" }

    %% C major for guitar, barred on third fret
    % verbose style
    % size 1.0
    % roman fret label, finger labels below string, straight barre
    c'1~\markup {
      % standard size
      \override #'(size . 1.0) {
        \override #'(fret-diagram-details . (
          (number-type . roman-lower)
          (finger-code . in-dot)
          (barre-type . straight))) {
          \fret-diagram-verbose #'(mute 6)
            (place-fret 5 3 1)
        }
      }
    }
  }
>>

```

```

        (place-fret 4 5 2)
        (place-fret 3 5 3)
        (place-fret 2 5 4)
        (place-fret 1 3 1)
        (barre 5 1 3))
    }
  }
}

%% C major for guitar, barred on third fret
% verbose style
% landscape orientation, arabic numbers, M for mute string
% no barre, fret label down or left, small mute label font
c'1^\markup {
  \override #'(fret-diagram-details . (
    (finger-code . below-string)
    (number-type . arabic)
    (label-dir . -1)
    (mute-string . "M")
    (orientation . landscape)
    (barre-type . none)
    (xo-font-magnification . 0.4)
    (xo-padding . 0.3))) {
    \fret-diagram-verbose #'((mute 6)
      (place-fret 5 3 1)
      (place-fret 4 5 2)
      (place-fret 3 5 3)
      (place-fret 2 5 4)
      (place-fret 1 3 1)
      (barre 5 1 3))
    }
  }
}

%% simple D chord
% terse style
% larger dots, centered dots, fewer frets
% label below string
d'1^\markup {
  \override #'(fret-diagram-details . (
    (finger-code . below-string)
    (dot-radius . 0.35)
    (dot-position . 0.5)
    (fret-count . 3))) {
    \fret-diagram-terse "x;x;o;2-1;3-2;2-3;"
  }
}
}
}
>>

```



Defining predefined fretboards for other instruments

Predefined fret diagrams can be added for new instruments in addition to the standards used for guitar. This file shows how this is done by defining a new string-tuning and a few predefined fretboards for the Venezuelan *cuatro*.

This file also shows how fingerings can be included in the chords used as reference points for the chord lookup, and displayed in the fret diagram and the `TabStaff`, but not the music.

These fretboards are not transposable because they contain string information. This is planned to be corrected in the future.

```
% add FretBoards for the Cuatro
% Note: This section could be put into a separate file
%     predefined-cuatro-fretboards.ly
%     and \included into each of your compositions
```

```
cuatroTuning = #`((ly:make-pitch 0 6 0)
                  ,(ly:make-pitch 1 3 SHARP)
                  ,(ly:make-pitch 1 1 0)
                  ,(ly:make-pitch 0 5 0))
```

```
dSix = { <a\4 b\1 d\3 fis\2> }
dMajor = { <a\4 d\1 d\3 fis \2> }
aMajSeven = { <a\4 cis\1 e\3 g\2> }
dMajSeven = { <a\4 c\1 d\3 fis\2> }
gMajor = { <b\4 b\1 d\3 g\2> }
```

```
\storePredefinedDiagram #default-fret-table \dSix
                        #cuatroTuning
                        #"o;o;o;o;"
\storePredefinedDiagram #default-fret-table \dMajor
                        #cuatroTuning
                        #"o;o;o;3-3;"
\storePredefinedDiagram #default-fret-table \aMajSeven
                        #cuatroTuning
                        #"o;2-2;1-1;2-3;"
\storePredefinedDiagram #default-fret-table \dMajSeven
                        #cuatroTuning
                        #"o;o;o;1-1;"
\storePredefinedDiagram #default-fret-table \gMajor
                        #cuatroTuning
                        #"2-2;o;1-1;o;"
```

```
% end of potential include file /predefined-cuatro-fretboards.ly
```

```
 #(set-global-staff-size 16)
```

```
 primerosNames = \chordmode {
```

```

    d:6 d a:maj7 d:maj7
  g
}
primeros = {
  \dSix \dMajor \aMajSeven \dMajSeven
  \gMajor
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \primerosNames
    }

    \new Staff {
      \new Voice \with {
        \remove "New_fingering_engraver"
      }
      \relative c'' {
        \primeros
      }
    }

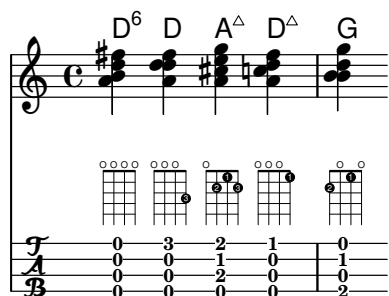
    \new FretBoards {
      \set Staff.stringTunings = #cuatroTuning
%      \override FretBoard
%        #'(fret-diagram-details string-count) = 4
      \override FretBoard.fret-diagram-details.finger-code = #'in-dot
      \primeros
    }

    \new TabStaff \relative c'' {
      \set TabStaff.stringTunings = #cuatroTuning
      \primeros
    }

  >>

  \layout {
    \context {
      \Score
      \override SpacingSpanner.base-shortest-duration =
        #(ly:make-moment 1 16)
    }
  }
  \midi { }
}

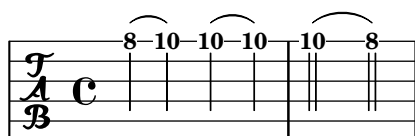
```

Faking a hammer in tablatures

A hammer in tablature can be faked with slurs.

```
\score {
  \new TabStaff {
    \relative c'' {
      \tabFullNotation
      c4( d) d( d)
      d2( c)
    }
  }
}
```

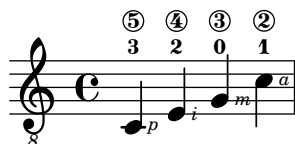


Fingerings, string indications, and right-hand fingerings

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, special notation is used:

- a *golpe* symbol to indicate a slap on the guitar body with the nail of the ring finger
- an arrow to indicate (the direction of) strokes
- different letters for fingering (“p”: thumb, “i”: index finger, “m”: middle finger, “a”: ring finger and “x”: little finger)
- 3- and 4-finger *rasgueados*; stroke upwards with all fingers, ending with an up- and down using the index finger

- *abanicos*: 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*: 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 '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
```

```
strokeDown = \markup\combine\arrow-head #Y #DOWN ##f \override #'(thickness . 1.3) \draw-li
```

```
% 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
```

```
% Just handy :)
```

```
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 '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\RHm>
  <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\RHx>16\rasgueado
    \headsOff
    <a e' a cis e\RHu\RH>
    <a e' a cis e\RHu\RHm>
    <a e' a cis e\RHu\RH>
    <a e' a cis e\RHd\RH>~
    \headsOn
  }
  <a e' a cis e>2
  r4
  \tupletsOff
  \tuplet 3/2 {
    <a e' a cis e\RHd\RHp>8\abanico
    \headsOff
  }
}

```

```

    <a e' a cis e\RHu\RHx>
    <a e' a cis e\RHu\RHx>
    \headsOn
}
\tuplet 3/2 {
    <a e' a cis e\RHd\RHp>8
    \headsOff
    <a e' a cis e\RHu\RHx>
    <a e' a cis e\RHu\RHx>
    \headsOn
}
\tuplet 3/2 {
    <a e' a cis e\RHd\RHp>8
    \headsOff
    <a e' a cis e\RHu\RHx>
    <a e' a cis e\RHu\RHx>
    \headsOn
}
\tuplet 3/2 {
    <a e' a cis e\RHd\RHp>8
    \headsOff
    <a e' a cis e\RHu\RHx>
    <a e' a cis e\RHu\RHx>
    \headsOn
}
\tupletsOff
\override Beam.positions = #'(2 . 2)
\tuplet 3/2 {
    a8\RHp\alzapua
    <e' a\RHu\RHg>
    <e a\RHd>
}
\tuplet 3/2 {
    a,8\RHp
    <e' a\RHu\RHg>
    <e a\RHd>
}
\tuplet 3/2 {
    a,8\RHp
    <e' a\RHu\RHg>
    <e a\RHd>
}
\tuplet 3/2 {
    a,8\RHp
    <e' a\RHu\RHg>
    <e a\RHd>
}
\tupletsOn
<a, e' a\RHu\RHm>1
\bar "|."
}

```

```

\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.

```

<<
  \chords {
    a2 a
    \repeat unfold 3 {
      c c c d d
    }
  }

  \new Voice = "mel" {

```

```

\textLengthOn
% 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

%% A chord for ukulele
a'2^\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;"
  }
}

%% 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'2^\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

%% C major for guitar, barred on third fret
% verbose style
% roman fret label, finger labels below string, straight barre
c'2^\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))
    }
  }
}

%% C major for guitar, barred on third fret
%% Double barre used to test barre function

```

```

% verbose style
c'2^\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))
    }
  }
}

%% C major for guitar, with capo on third fret
% verbose style
c'2^\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))
    }
  }
}

%% simple D chord
d'2^\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;"
  }
}

%% simple D chord, large top fret thickness
d'2~\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

%% C major for guitar, barred on third fret
% verbose style
% roman fret label, finger labels below string, straight barre
c'2~\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))
    }
  }
}

%% C major for guitar, barred on third fret
%% Double barre used to test barre function
% verbose style
c'2~\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))
    }
}
}

%% C major for guitar, with capo on third fret
% verbose style
c'2^\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))
    }
  }
}

%% simple D chord
d'2^\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;"
  }
}

```

```

    }
}

%% simple D chord, large top fret thickness
d'2^\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

%% C major for guitar, barred on third fret
% verbose style
% roman fret label, finger labels below string, straight barre
c'2^\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))
    }
  }
}

%% C major for guitar, barred on third fret
%% Double barre used to test barre function
% verbose style
c'2^\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))
    }
}
}

%% C major for guitar, with capo on third fret
% verbose style
c'2^\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))
    }
  }
}

%% simple D chord
d'2^\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;"
  }
}

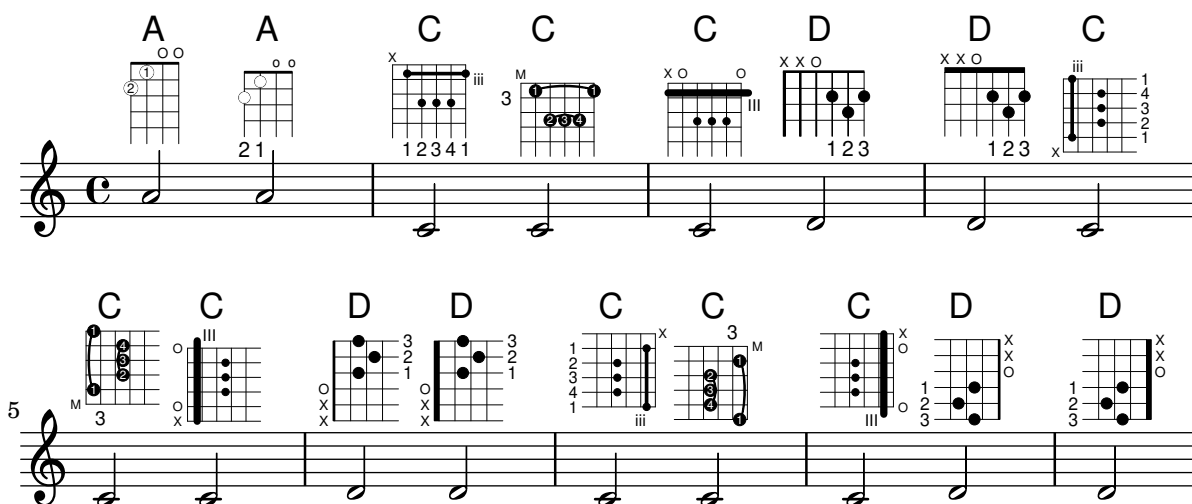
%% simple D chord, large top fret thickness
d'2^\markup {
  \override #'(fret-diagram-details . (

```

```

(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;"
}
}
}
>>

```



Fretboards alternate tables

Alternate fretboard tables can be created. These would be used in order to have alternate fretboards for a given chord.

In order to use an alternate fretboard table, the table must first be created. Fretboards are then added to the table.

The created fretboard table can be blank, or it can be copied from an existing table.

The table to be used in displaying predefined fretboards is selected by the property `\predefinedDiagramTable`.

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

```
% Make a blank new fretboard table
#(define custom-fretboard-table-one
  (make-fretboard-table))
```

```
% Make a new fretboard table as a copy of default-fret-table
#(define custom-fretboard-table-two
  (make-fretboard-table default-fret-table))
```

```
% Add a chord to custom-fretboard-table-one
\storePredefinedDiagram #custom-fretboard-table-one
  \chordmode {c}
  #guitar-tuning
  "3-(;3;5;5;5;3-);"
```

```

% Add a chord to custom-fretboard-table-two
\storePredefinedDiagram #custom-fretboard-table-two
    \chordmode {c}
    #guitar-tuning
    "x;3;5;5;5;o;"

<<
\chords {
  c1 | d1 |
  c1 | d1 |
  c1 | d1 |
}
\new FretBoards {
  \chordmode {
    \set predefinedDiagramTable = #default-fret-table
    c1 | d1 |
    \set predefinedDiagramTable = #custom-fretboard-table-one
    c1 | d1 |
    \set predefinedDiagramTable = #custom-fretboard-table-two
    c1 | d1 |
  }
}
\new Staff {
  \clef "treble_8"
  <<
    \chordmode {
      c1 | d1 |
      c1 | d1 |
      c1 | d1 |
    }
    {
      s1_\markup "Default table" | s1 |
      s1_\markup \column {"New table" "from empty"} | s1 |
      s1_\markup \column {"New table" "from default"} | s1 |
    }
  >>
}
>>

```

Default table New table from empty New table from default

Fretted-string harmonics in tablature

Demonstrates fretted-string harmonics in tablature

```
pinchedHarmonics = {
```

```

\textSpannerDown
\override TextSpanner.bound-details.left.text =
  \markup { \halign #-0.5 \teeny "PH" }
  \override TextSpanner.style =
    #'dashed-line
\override TextSpanner.dash-period = #0.6
\override TextSpanner.bound-details.right.attach-dir = #1
\override TextSpanner.bound-details.right.text =
  \markup { \draw-line #'(0 . 1) }
\override TextSpanner.bound-details.right.padding = #-0.5
}

harmonics = {
  %artificial harmonics (AH)
  \textLengthOn
  <\parenthesize b b'\harmonic>4_\markup { \teeny "AH 16" }
  <\parenthesize g g'\harmonic>4_\markup { \teeny "AH 17" }
  <\parenthesize d' d'\harmonic>2_\markup { \teeny "AH 19" }
  %pinched harmonics (PH)
  \pinchedHarmonics
  <a'\harmonic>2\startTextSpan
  <d'\harmonic>4
  <e'\harmonic>4\stopTextSpan
  %tapped harmonics (TH)
  <\parenthesize g\4 g'\harmonic>4_\markup { \teeny "TH 17" }
  <\parenthesize a\4 a'\harmonic>4_\markup { \teeny "TH 19" }
  <\parenthesize c'\3 c'\harmonic>2_\markup { \teeny "TH 17" }
  %touch harmonics (TCH)
  a4( <e'\harmonic>2. )_\markup { \teeny "TCH" }
}

frettedStrings = {
  %artificial harmonics (AH)
  \harmonicByFret #4 g4\3
  \harmonicByFret #5 d4\4
  \harmonicByFret #7 g2\3
  %pinched harmonics (PH)
  \harmonicByFret #7 d2\4
  \harmonicByFret #5 d4\4
  \harmonicByFret #7 a4\5
  %tapped harmonics (TH)
  \harmonicByFret #5 d4\4
  \harmonicByFret #7 d4\4
  \harmonicByFret #5 g2\3
  %touch harmonics (TCH)
  a4 \harmonicByFret #9 g2.\3
}

\score {
  <<
  \new Staff
  \with { \omit StringNumber } {

```

```

\new Voice {
  \clef "treble_8"
  \harmonics
}
}
\new TabStaff {
  \new TabVoice {
    \frettedStrings
  }
}
>>
}

```

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 \hide Stem
  \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
    }
  >>
}

```

The image shows a musical score for guitar. The top staff is a treble clef with a common time signature. It contains a melody with notes on the 8th, 5th, 4th, and 3rd frets. Above the notes are fingerings: 2, 2, 2, 5, 4, 3. Slurs indicate 'Slide into' and 'Slide from'. The bottom staff shows fret diagrams for the guitar strings, with fret numbers 8, 3, 7, 4, and 8 indicated.

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.
}
}
>>

```

Chord diagrams for C, F, G, and C are shown above the guitar staff. The guitar staff is in C major, and the voice staff is in C major. The lyrics are: This is my song. I like to sing.

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>)
  }
}

```

Diagram showing a hammer-on and pull-off using chords. The notation includes a single arc connecting the notes G and B, with fret numbers 7-8-7 for G and 8-10-8 for 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
  }
}

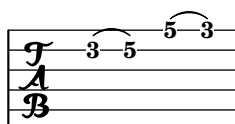
```

Diagram showing a hammer-on and pull-off using voices. The notation includes a single arc connecting the notes G and B, with fret numbers 0-2 for G and 0-2 for B.

Hammer on and pull off

Hammer-on and pull-off can be obtained using slurs.

```
\new TabStaff {
  \relative c' {
    d4( e\2)
    a( g)
  }
}
```



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 `#'padding` or `#'extra-offset` values (as shown in the first snippet)
- 2) you can add an invisible voice and attach the fret diagrams to the invisible notes in that voice (as shown in the second example).

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
  % THE FOLLOWING IS THE COMMAND TO MOVE THE CHORD NAME
  \once \override ChordNames.ChordName.extra-offset = #'(10 . 0)
  b8:13 s2.
  % THIS LINE IS THE SECOND METHOD
  s4 s4 b4:13
}

\score
{
  <<
    \new ChordNames \harmonies
    \new Staff
    {a8^\markup { \fret-diagram "6-x;5-0;4-2;3-0;2-0;1-2;" }
  % THE FOLLOWING IS THE COMMAND TO MOVE THE FRET DIAGRAM
  \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
  % HERE IS THE SECOND METHOD
  <<
    { a8 b4.~ b4. a8}
    { s4 s4 s4^\markup { \fret-diagram "6-x;5-2;4-4;3-2;2-2;1-4;" }
    }
  >>
  }
  >>
```

}

The image shows two staves of musical notation for fretted strings. The first staff is in treble clef with a common time signature (C). It contains a melody with a slur connecting two notes. Above the staff are two guitar chord diagrams: A^{9 13} and B^{9 13}. The second staff is a second system, starting with a '2' indicating a second ending. It also contains a melody with a slur connecting two notes. Above the staff is a B^{9 13} guitar chord diagram.

Jazz combo template

This is quite an advanced template, for a jazz ensemble. Note that all instruments are notated in `\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 in the documentation
\paper {
  paper-width = 130
}

%#(set-global-staff-size 16)
\include "english.ly"

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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
    \nsl
}
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
    \nsl
    c1
}
gtrHarmony = \chordmode {
    \jazzChords
    s1 c2:min7+ d2:maj9
}
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 %%%%%%%%%%%

\score {
  <<
    \new StaffGroup = "horns" <<
      \new Staff = "trumpet" \with { instrumentName = "Trumpet" }
      \trumpet
      \new Staff = "altosax" \with { instrumentName = "Alto Sax" }
      \altoSax
      \new ChordNames = "barichords" \with { instrumentName = "Trumpet" }
      \bariHarmony
      \new Staff = "barisax" \with { instrumentName = "Bari Sax" }
      \bariSax
      \new Staff = "trombone" \with { instrumentName = "Trombone" }
      \trombone
    >>

    \new StaffGroup = "rhythm" <<
      \new ChordNames = "chords" \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^Δ Solo C[♯]m⁷

Cm^Δ D^{Δ9}

Laissez vibrer ties

Laissez vibrer ties have a fixed size. Their formatting can be tuned using 'tie-configuration.

```
\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 usually the first string is printed topmost. If you want to have it at the bottom change the `stringOneTopmost`-context-property. For a context-wide setting this could be done in `layout` 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
  }
>>
```

The image displays a musical score with two staves. The top staff is a treble clef with a C-clef (soprano) and a key signature of one sharp (F#). It contains two measures of music. The first measure is labeled 'default' and the second is labeled 'italian (historic)'. The bottom staff is a bass clef with a C-clef (bass) and a key signature of one sharp (F#). It contains two measures of music, corresponding to the top staff. The first measure is labeled 'default' and the second is labeled '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
      }
    }
  }
}

```

}
>>
}

8 1st harm. 2nd harm. 3rd harm.

12 12 7 7 19 19 5 5 24 24

6 4th harm. 5th harm.

4 4 9 9 16 16 3 3

10 6th harm. 7th harm. 8th harm.

2.7 2.7 2.3 2.3 2 2

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
}
```



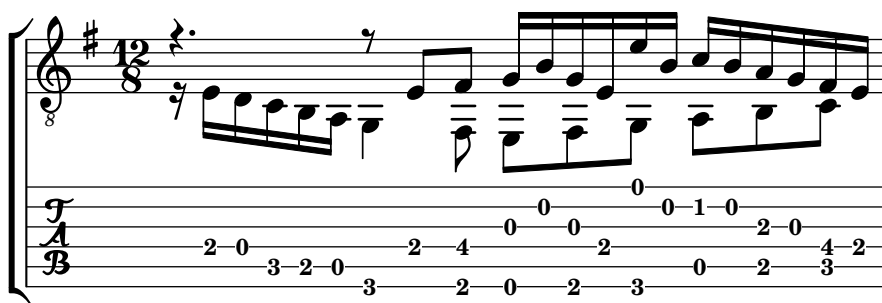
Polyphony in tablature

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
      >>
    >>
  >>
}
```



Slides in tablature

Slides can be typeset in both `Staff` and `TabStaff` contexts:

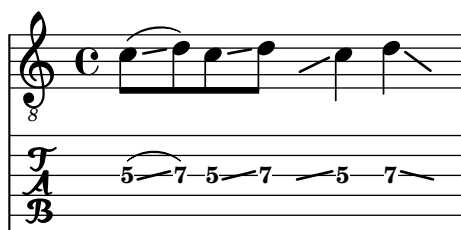
```
slides = {
  c'8\3(\glissando d'8\3)
  c'8\3\glissando d'8\3
  \hideNotes
}
```

```

\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
    }
  }
}

```



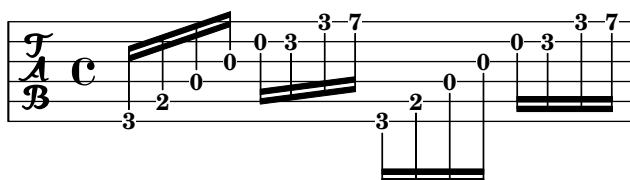
Stem and beam behavior in tablature

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
  }
}

```

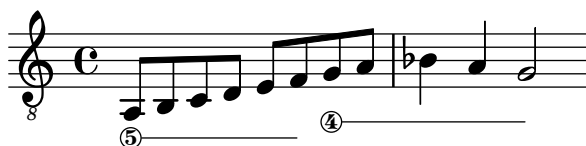


String number extender lines

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"
  \stringNumberSpanner "5"
  \textSpannerDown
  a8\startTextSpan
  b c d e f\stopTextSpan
  \stringNumberSpanner "4"
  g\startTextSpan a
  bes4 a g2\stopTextSpan
}
```



Unfretted strings

Section “Unfretted string instruments” in *Notation Reference*

Creating slurs across voices

In some situations, it may be 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 2nd 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`.

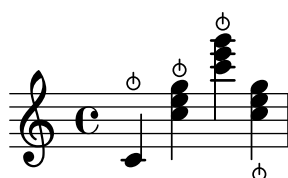
```
\relative c''' {
  \time 3/4
  \key f \major
  \set harmonicDots = ##t
  <bes f'\harmonic>2. ~
  <bes f'\harmonic>4. <a e'\harmonic>8( <gis dis'\harmonic> <g d'\harmonic>)
  <fis cis'\harmonic>2.
  <bes f'\harmonic>2.
}
```



Snap-pizzicato or 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

```
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” snippet produces a nice string quartet, but what if you needed to print parts? This new template demonstrates how to use the `\tag` feature to easily split a piece into individual parts.

You need to split this template into separate files; the filenames 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 appropriate part.

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 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 "|."
  }
}
```

```
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 { }  
}  
  
%}
```

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 and Violin 2 are in treble clef, Viola is in alto clef, and Cello is in bass clef. The notes are as follows:

Instrument	Measure 1	Measure 2
Violin 1	C4, D4, E4	F4, G4, A4
Violin 2	C4, D4, E4	F4, G4, A4
Viola	C3, D3, E3	F3, G3, A3
Cello	C2, D2, E2	F2, G2, A2

Winds

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.

```
centermarkup = {
  \once \override TextScript.self-alignment-X = #CENTER
  \once \override TextScript.X-offset =#(lambda (g)
    (+ (ly:self-alignment-interface::centered-on-x-parent g)
      (ly:self-alignment-interface::x-aligned-on-self g)))
}
```

```

}

\score {
  \relative c' {
    g\open
    \once \override TextScript.staff-padding = #-1.0
    \centermarkup
    g^\markup {
      \combine
      \musicglyph "scripts.open"
      \musicglyph "scripts.tenuto"
    }
    \centermarkup
    g^\markup {
      \combine
      \musicglyph "scripts.open"
      \musicglyph "scripts.stopped"
    }
    g\stopped
  }
}

```



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.

```

slap =
#(define-music-function (music) (ly:music?)
#{
  \temporary \override NoteHead.stencil =
  #(lambda (grob)
    (grob-interpret-markup grob
      (markup #:musicglyph "scripts.sforzato")))
  \temporary \override NoteHead.stem-attachment =
  #(lambda (grob)
    (let* ((thickness (ly:staff-symbol-line-thickness grob))
      (stem (ly:grob-object grob 'stem))
      (dir (ly:grob-property stem 'direction UP)))
      (cons 1 (+ (if (= dir DOWN)
        0.5
        0)
        (/ thickness 2))))))
  #music
  \revert NoteHead.stencil
  \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

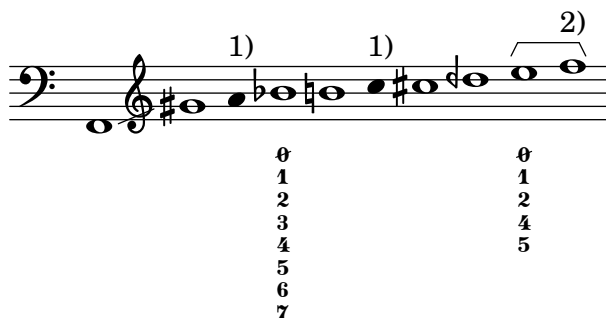
```
centermarkup = {
  \once \override TextScript.self-alignment-X = #CENTER
  \once \override TextScript.X-offset = #(lambda (g)
    (+ (ly:self-alignment-interface::centered-on-x-parent g)
      (ly:self-alignment-interface::x-aligned-on-self g)))
}
```

```

}

\score {
  \new Staff \with {
    \remove "Time_signature_engraver"
    \omit Stem
    \omit Flag
    \consists "Horizontal_bracket_engraver"
  }
  {
    \clef bass
    \set Score.timing = ##f
    f,1*1/4 \glissando
    \clef violin
    gis'1*1/4
    \stemDown a'4^\markup {1)}
    \centermarkup
    \once \override TextScript.padding = #2
    bes'1*1/4_\markup {\override #'(baseline-skip . 1.7) \column
      { \fontsize #-5 \slashed-digit #0 \finger 1 \finger 2
        \finger 3 \finger 4 \finger 5 \finger 6 \finger 7} }
    b'1*1/4
    c''4^\markup {1)}
    \centermarkup
    \once \override TextScript.padding = #2
    cis''1*1/4
    deh''1*1/4
    \centermarkup
    \once \override TextScript.padding = #2
    \once \override Staff.HorizontalBracket.direction = #UP
    e''1*1/4_\markup {\override #'(baseline-skip . 1.7) \column
      { \fontsize #-5 \slashed-digit #0 \finger 1 \finger 2
        \finger 4 \finger 5} }\startGroup
    f''1*1/4^\markup {2)}\stopGroup
  }
}

```



Woodwind diagrams key lists

The snippet below produces a list of all possible keys and key settings for woodwind diagrams as defined in `scm/define-woodwind-diagrams.scm`. The list will be displayed in the log file,

but not in the music. If output to the console is wanted, omit the `(current-error-port)` from the commands.

```
#(print-keys-verbose 'piccolo (current-error-port))
#(print-keys-verbose 'flute (current-error-port))
#(print-keys-verbose 'flute-b-extension (current-error-port))
#(print-keys-verbose 'tin-whistle (current-error-port))
#(print-keys-verbose 'oboe (current-error-port))
#(print-keys-verbose 'clarinet (current-error-port))
#(print-keys-verbose 'bass-clarinet (current-error-port))
#(print-keys-verbose 'low-bass-clarinet (current-error-port))
#(print-keys-verbose 'saxophone (current-error-port))
#(print-keys-verbose 'soprano-saxophone (current-error-port))
#(print-keys-verbose 'alto-saxophone (current-error-port))
#(print-keys-verbose 'tenor-saxophone (current-error-port))
#(print-keys-verbose 'baritone-saxophone (current-error-port))
#(print-keys-verbose 'bassoon (current-error-port))
#(print-keys-verbose 'contrabassoon (current-error-port))
```

```
\score {c'1}
```



Woodwind diagrams listing

The following music shows all of the woodwind diagrams currently defined in LilyPond.

```
\layout {
  indent = 0
}

\relative c' {
  \textLengthOn
  c1~
  \markup {
    \center-column {
      'tin-whistle
      " "
      \woodwind-diagram
      #'tin-whistle
      #'()
    }
  }

  c1~
  \markup {
    \center-column {
      'piccolo
      " "
      \woodwind-diagram
      #'piccolo
      #'()
    }
  }
}
```

```

    }
  }

  c1~
  \markup {
    \center-column {
      'flute
      " "

      \woodwind-diagram
      #'flute
      #'()
    }
  }

  c1~\markup {
    \center-column {
      'oboe
      " "

      \woodwind-diagram
      #'oboe
      #'()
    }
  }

  c1~\markup {
    \center-column {
      'clarinet
      " "

      \woodwind-diagram
      #'clarinet
      #'()
    }
  }

  c1~\markup {
    \center-column {
      'bass-clarinet
      " "

      \woodwind-diagram
      #'bass-clarinet
      #'()
    }
  }

  c1~\markup {
    \center-column {
      'saxophone
      " "

      \woodwind-diagram
      #'saxophone
      #'()
    }
  }

```

```

c1^\markup {
  \center-column {
    'bassoon
    " "
    \woodwind-diagram
    #'bassoon
    #'()
  }
}

```

```

c1^\markup {
  \center-column {
    'contrabassoon
    " "
    \woodwind-diagram
    #'contrabassoon
    #'()
  }
}
}

```

The image displays two systems of musical staves, each with a treble clef and a common time signature 'C'. The first system contains five staves, each with a woodwind instrument diagram above it: 'tin-whistle, 'piccolo, 'flute, 'oboe, and 'clarinet. The second system contains four staves: 'bass-clarinet, 'saxophone, 'bassoon, and 'contrabassoon. The diagrams are stylized line drawings of the instruments, showing their unique shapes and key mechanisms. The staves are empty, with only the instrument diagrams and the time signature visible.

Ancient notation

Section “Ancient notation” in *Notation Reference*

Adding a figured bass above or below the notes

When writing a figured bass, you can place the figures above or below the bass notes, by defining the `BassFigureAlignmentPositioning.direction` property (exclusively in a `Staff` context). Choices are `#UP` (or `#1`), `#CENTER` (or `#0`) and `#DOWN` (or `#-1`).

This property can be changed as many times as you wish. Use `\once \override` if you don’t want the override to apply to the whole score.

```
bass = {
  \clef bass
  g4 b, c d
  e d8 c d2
}

continuo = \figuremode {
  <_>4 <6>4 <5/>4
  \override Staff.BassFigureAlignmentPositioning.direction = #UP
  %\bassFigureStaffAlignmentUp
  <_+ >4 <6>
  \set Staff.useBassFigureExtenders = ##t
  \override Staff.BassFigureAlignmentPositioning.direction = #DOWN
  %\bassFigureStaffAlignmentDown
  <4>4. <4>8 <_+>4
}

\score {
  <<
    \new Staff = bassStaff \bass
    \context Staff = bassStaff \continuo
  >>
}
```



Ancient fonts

Shown here are many of the symbols that are included in LilyPond’s ancient notation.

```
upperStaff = \new VaticanaStaff = "upperStaff" <<
  \context VaticanaVoice <<
    \transpose c c {

      \override NoteHead.style = #'vaticana.punctum
      \key es \major
      \clef "vaticana-fa2"
      c1 des e f ges
```

```

\override NoteHead.style = #'vaticana.inclinatum
a! b ces'
\bar "|"

\override NoteHead.style = #'vaticana.quilisma
b! des'! ges! fes!
\breath
\clef "vaticana-fa1"
\override NoteHead.style = #'vaticana.plica
es d
\override NoteHead.style = #'vaticana.reverse.plica
c d
\bar "|"

\override NoteHead.style = #'vaticana.punctum.cavum
es f
\override NoteHead.style = #'vaticana.lpes
g as
\override NoteHead.style = #'vaticana.upes
bes as
\override NoteHead.style = #'vaticana.vupes
g f
\override NoteHead.style = #'vaticana.linea.punctum
\once \override Staff.BarLine.bar-extent = #'(-1 . 1) \bar "|"

es d
\override NoteHead.style = #'vaticana.epiphonus
c d
\override NoteHead.style = #'vaticana.cephalicus
es f

\override Staff.KeySignature.glyph-name-alist =
  #alteration-medicaea-glyph-name-alist
\override Staff.Accidental.glyph-name-alist =
  #alteration-medicaea-glyph-name-alist
\override Staff.Custos.style = #'medicaea
\override NoteHead.style = #'medicaea.punctum
\clef "medicaea-fa2"
ces des
\bar "|"

e! f! ges
\clef "medicaea-do2"
\override NoteHead.style = #'medicaea.inclinatum
a! b! ces'
\override NoteHead.style = #'medicaea.virga
b! a!
\bar "|"

ges fes
\clef "medicaea-fa1"

```

```

\override NoteHead.style = #'medicaea.rvirga
e des ces

\override Staff.KeySignature.glyph-name-alist =
  #alteration-hufnagel-glyph-name-alist
\override Staff.Accidental.glyph-name-alist =
  #alteration-hufnagel-glyph-name-alist
\override Staff.Custos.style = #'hufnagel
\override NoteHead.style = #'hufnagel.punctum
\clef "hufnagel-fa2"
ces des es
\bar "|"

fes ges
\clef "hufnagel-do2"
\override NoteHead.style = #'hufnagel.lpes
as! bes! ces'
\override NoteHead.style = #'hufnagel.virga
bes! as!
\bar "|"

ges! fes!
\clef "hufnagel-do-fa"
\override NoteHead.style = #'hufnagel.punctum
es! des ces des! es! fes!
\bar "||"

s32*1
}
>>
>>

lowerStaff = \new MensuralStaff = "lowerStaff" <<
  \context MensuralVoice <<
    \transpose c c {

      \key a \major
      cis'1 d'\breve gis'\breve e'\breve \[ e'\longa fis'\longa \]
      \set Staff.forceClef = ##t
      \clef "neomensural-c2"
      cis1
      \bar "|"

      \[ g\breve dis''\longa \]
      b\breve \[ a\longa d\longa \]
      \clef "petrucci-c2"

      fis1 ces1
      \clef "petrucci-c2"
      r\longa
      \set Staff.forceClef = ##t
      \clef "mensural-c2"
    }
  }

```



```

r\breve
\bar "|"

r2
\clef "mensural-g"
r4 r8 r16 r16
\override NoteHead.style = #'mensural
\override Rest.style = #'mensural
\clef "petrucci-f"
c8 b, c16 b, c32 b, c64 b, c64 b,
d8 e d16 e d32 e d64 e d64 e
r\longa
\set Staff.forceClef = ##t
\clef "petrucci-f"
r\breve
\bar "|"

r\breve
\clef "mensural-f"
r2 r4 r8 r16 r16

\set Staff.forceClef = ##t
\clef "mensural-f"
e\breve f g a1
\clef "mensural-g"

\[ bes'!\longa a'!\longa c'!\longa \]
e'1 d' c' d' \bar "|"
\bar "|"

bes'!\longa fis'!1 as'!1 ges'!\longa % lig
\set Staff.forceClef = ##t
\clef "mensural-g"
e'2 d' c' \bar "|"

\set Staff.forceClef = ##t
\clef "petrucci-g"
c'2 d' e' f'
\clef "petrucci-g"
g' as'! bes'! cis'!
bes'! as'! gis'! fis'!
\set Staff.forceClef = ##t
\clef "mensural-g"
es'! des'! cis'!1 \bar "||"
}
>>
>>

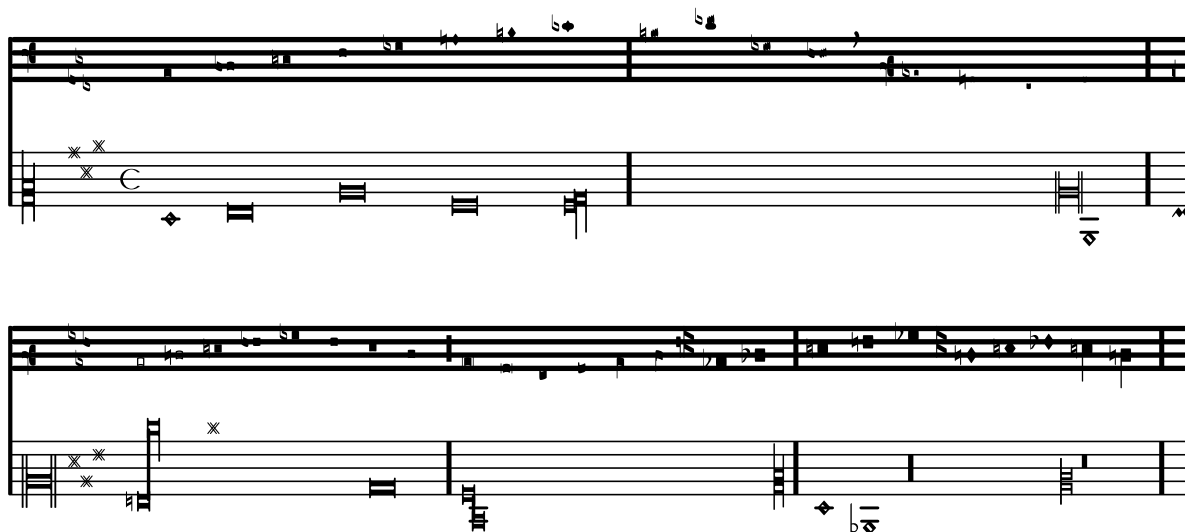
\paper {
  line-thickness = #(/ staff-space 5.0)
}

```

```

\score {
  <<
    \upperStaff
    \lowerStaff
  >>
  \layout {
    indent = 0.0
    \context {
      \Score
      timing = ##f
    }
    \context {
      \MensuralVoice
      \override NoteHead.style = #'neomensural
      \override Rest.style = #'neomensural
      \override Flag.style = #'mensural
      \override Stem.thickness = #1.0
    }
    \context {
      \MensuralStaff
      \revert BarLine.transparent
      \override KeySignature.glyph-name-alist =
        #alteration-mensural-glyph-name-alist
      clefGlyph = #"clefs.petrucchi.c2"
    }
    \context {
      \VaticanaStaff
      \revert BarLine.transparent
      \override StaffSymbol.thickness = #2.0
      \override KeySignature.glyph-name-alist =
        #alteration-vaticana-glyph-name-alist
      \override Custos.neutral-position = #4
    }
  }
}

```





Ancient notation template – modern transcription of gregorian music

This example demonstrates how to do modern transcription of Gregorian music. Gregorian music has no measure, no stems; it uses only half and quarter note heads, and special marks, indicating rests of different length.

```
\include "gregorian.ly"
```

```
chant = \relative c' {
  \set Score.timing = ##f
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g) a2 \finalis
}

verba = \lyricmode {
  Lo -- rem ip -- sum do -- lor sit a -- met
}

\score {
  \new Staff <<
    \new Voice = "melody" \chant
    \new Lyrics = "one" \lyricsto melody \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Bar_engraver"
      \hide Stem
    }
    \context {
      \Voice
      \override Stem.length = #0
    }
    \context {
```

```

\Score
barAlways = ##t
}
}
}

```



Ancient time signatures

Time signatures may also be engraved in an old style.

```

{
\override Staff.TimeSignature.style = #'neomensural
s1
}

```



Chant or psalms notation

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

```

stemOff = \hide Staff.Stem
stemOn = \undo \stemOff

```

```

\score {
\new Staff \with { \remove "Time_signature_engraver" }
{
\key g \minor
\cadenzaOn
\stemOff a'\breve bes'4 g'4
\stemOn a'2 \bar "||"
\stemOff a'\breve g'4 a'4
\stemOn f'2 \bar "||"
\stemOff a'\breve^{\markup { \italic flexe }}
\stemOn g'2 \bar "||"
}
}

```



Custodes

Custodes may be engraved in various styles.

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

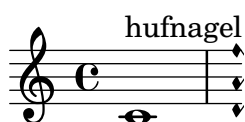
\new Staff \with { \consists "Custos_engraver" } \relative c' {
  \override Staff.Custos.neutral-position = #4

  \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. Musicians today are used to bar lines, but these were not known during the period of mensural music. As a compromise, bar lines are often printed between the staves, a layout style called mensurstriche layout.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A short excerpt from the Jubilate Deo by Orlando 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 = {
  \clef "neomensural-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 = {
  \clef "neomensural-c3"
  \key f \major
  \time 2/2
  r1 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 = {
  \clef "neomensural-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 = {
    \clef "mensural-f"
    \key f \major
    \time 2/2
    r\maxima
    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 \discantusIncipit
                \global
                \discantusNotes
            >>
            \new Lyrics \lyricsto discantusNotes { \discantusLyrics }
            \new Voice = "altusNotes" <<
                \set Staff.instrumentName = "Altus"
                \global
                \incipit \altusIncipit
                \altusNotes
            >>
            \new Lyrics \lyricsto altusNotes { \altusLyrics }
            \new Voice = "tenorNotes" <<
                \set Staff.instrumentName = "Tenor"

```



```

        \global
        \incipit \tenorIncipit
        \tenorNotes
    >>
    \new Lyrics \lyricsto tenorNotes { \tenorLyrics }
    \new Voice = "bassusNotes" <<
        \set Staff.instrumentName = "Bassus"
        \global
        \incipit \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 = 6\cm
    incipit-width = 4\cm
}
}

```

Discantus

Altus

Tenor

Bassus

Ju - bi - la - te De -

Ju bi - la - te

o, om - nis ter - ra, om - ... -us.

De - o, om - nis ter - ra, ... -us.

Ju - bi - la - te ... -us.

Ju - bi - ... -us.

Mensurstriche layout (bar lines between the staves)

The mensurstriche-layout where the bar lines do not show on the staves but between staves can be achieved with a **StaffGroup** instead of a **ChoirStaff**. The bar line on staves is blanked out using `\hide`.

```
global = {
  \hide Staff.BarLine
  s1 s
  % the final bar line is not interrupted
  \undo \hide Staff.BarLine
  \bar "|."
}

\new StaffGroup \relative c'' {
  <<
    \new Staff { << \global { c1 c } >> }
    \new Staff { << \global { c c } >> }
  >>
}
```



Rest styles

Rests may be used in various styles.

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

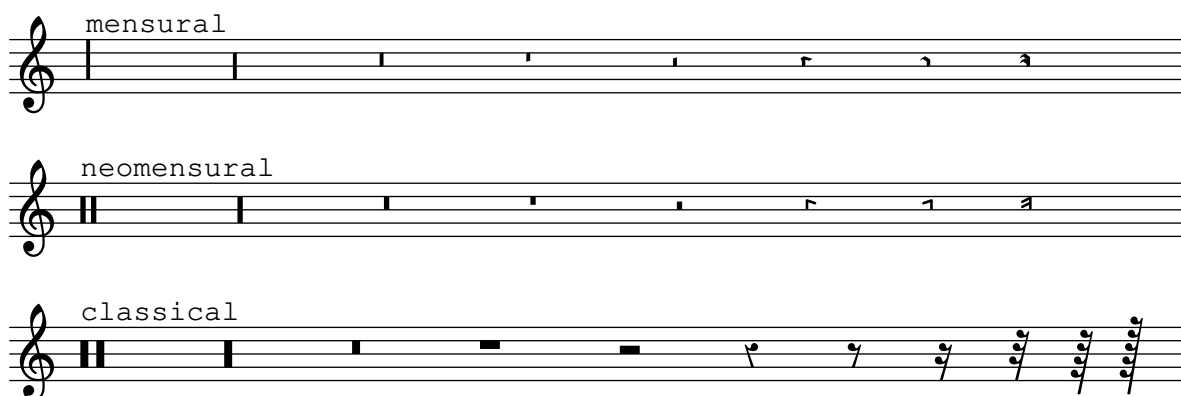
  \override Staff.Rest.style = #'mensural
  r\maxima^\markup \typewriter { mensural }
  r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
  \bar ""
  \break

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

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

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

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





Using tags to produce mensural and modern music from the same source

By using tags, it's possible to use the same music to produce both mensural and modern music. In this snippet, a function `menrest` is introduced, allowing mensural rests to be pitched as in the original, but with modern rests in the standard staff position. Tags are used to produce different types of bar line at the end of the music, but tags can also be used where other differences are needed: for example using “whole measure rests” (`R1`, `R\breve` etc.) in modern music, but normal rests (`r1`, `r\breve`, etc.) in the mensural version. Note that converting mensural music to its modern equivalent is usually referred to as **transcription**.

```
menrest = #(define-music-function (note)
  (ly:music?)
  #{
    \tag #'mens $(make-music 'RestEvent note)
    \tag #'mod $(make-music 'RestEvent note 'pitch '())
  })

MensStyle = {
  \autoBeamOff
  \override NoteHead.style = #'petrucci
  \override Score.BarNumber.transparent = ##t
  \override Stem.neutral-direction = #up
}

finalis = {
  \once \override BreathingSign.stencil = #ly:breathing-sign::finalis
  \once \override BreathingSign.Y-offset = #0
  \once \override BreathingSign.minimum-X-extent = #'(-1.0 . 0.0)
  \once \override BreathingSign.minimum-Y-extent = #'(-2.5 . 2.5)

  \breathe
}

Music = \relative c'' {
  \set Score.tempoHideNote = ##t
  \key f \major
  \time 4/4
  g1 d'2 \menrest bes4 bes2 a2 r4 g4 fis2.
  \tag #'mens { \finalis }
  \tag #'mod { \bar "||" }
}

MenLyr = \lyricmode { So farre, deere life, deare life }
ModLyr = \lyricmode { So far, dear life, dear life }

\score {
```

```

\keepWithTag #'mens {
  <<
    \new MensuralStaff
    {
      \new MensuralVoice = Cantus
      \clef "mensural-c1" \MensStyle \Music
    }
    \new Lyrics \lyricsto Cantus \MenLyr
  >>
}
}

\score {
  \keepWithTag #'mod {
    \new ChoirStaff <<
      \new Staff
      {
        \new Voice = Sop \with {
          \remove "Note_heads_engraver"
          \consists "Completion_heads_engraver"
          \remove "Rest_engraver"
          \consists "Completion_rest_engraver" }
        {
          \shiftDurations #1 #0 { \autoBeamOff \Music }
        }
      }
      \new Lyrics \lyricsto Sop \ModLyr
    >>
  }
}

```



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  
}
```



World music

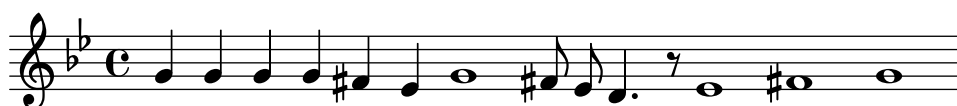
Section “World music” in *Notation Reference*

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 will cause the accidental to be marked only once. Here is an example of what could be the start of a *hijaz* improvisation:

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

```
\relative sol' {
  \key re \kurd
  \accidentalStyle forget
  \cadenzaOn
  sol4 sol sol sol fad mib sol1 fad8 mib re4. r8 mib1 fad sol
}
```



Makam example

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

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

```
% Initialize makam settings
```

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

```
\relative c' {
  \set Staff.keyAlterations = #`((6 . ,(- KOMA)) (3 . ,BAKIYE))
  c4 cc db fk
  gbm4 gfc gfb efk
  fk4 db cc c
}
```



Non-traditional key signatures

The commonly used `\key` command sets the `keyAlterations` property in the `Staff` context. To create non-standard key signatures, set this property directly.

The format of this command is a list:

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

where, for each element in the list `octave` specifies the octave (0 being the octave from middle c to the b above), `step` specifies the note within the octave (0 means c and 6 means b), and `alter` is `,SHARP`, `,FLAT`, `,DOUBLE-SHARP` etc.

Alternatively, using the more concise format for each item in the list, (**step . alter**) specifies the same alteration holds in all octaves. For microtonal scales where a “sharp” is not 100 cents, **alter** refers to the 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))

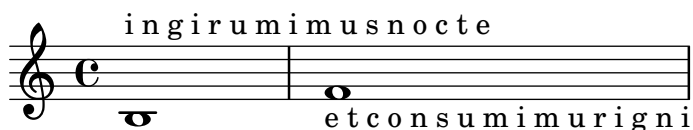
  %\set Staff.extraNatural = ##f
  re reb \down reb resd
  dod dob dosd \down dob |
  dobsb dodsdo do do |
}
```



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.

```
% Initialize makam settings
\include "turkish-makam.ly"

\header {
  title = "Hüseyini Saz Semaisi"
  composer = "Lavtacı Andon"
}

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



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

```
\key a \huseyni
```

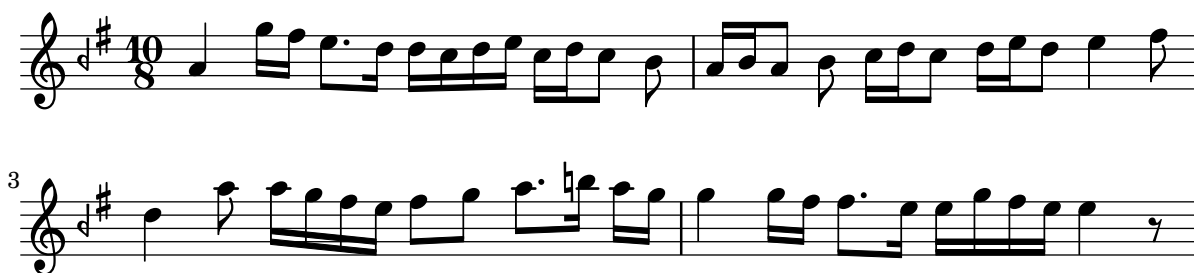
```
\time 10/8
```

```
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 |
```

```
}
```

Hüseyini Saz Semaisi

Lavtacı Andon



Contexts and engravers

Section “Changing defaults” in *Notation Reference*

Section “Contexts and engravers” in *Learning Manual*

Adding a figured bass above or below the notes

When writing a figured bass, you can place the figures above or below the bass notes, by defining the `BassFigureAlignmentPositioning.direction` property (exclusively in a `Staff` context). Choices are `#UP` (or `#1`), `#CENTER` (or `#0`) and `#DOWN` (or `#-1`).

This property can be changed as many times as you wish. Use `\once \override` if you don’t want the override to apply to the whole score.

```
bass = {
  \clef bass
  g4 b, c d
  e d8 c d2
}

continuo = \figuremode {
  <_>4 <6>4 <5/>4
  \override Staff.BassFigureAlignmentPositioning.direction = #UP
  %\bassFigureStaffAlignmentUp
  <_+ >4 <6>
  \set Staff.useBassFigureExtenders = ##t
  \override Staff.BassFigureAlignmentPositioning.direction = #DOWN
  %\bassFigureStaffAlignmentDown
  <4>4. <4>8 <_+>4
}

\score {
  <<
    \new Staff = bassStaff \bass
    \context Staff = bassStaff \continuo
  >>
}
```



Adding an extra staff at a line break

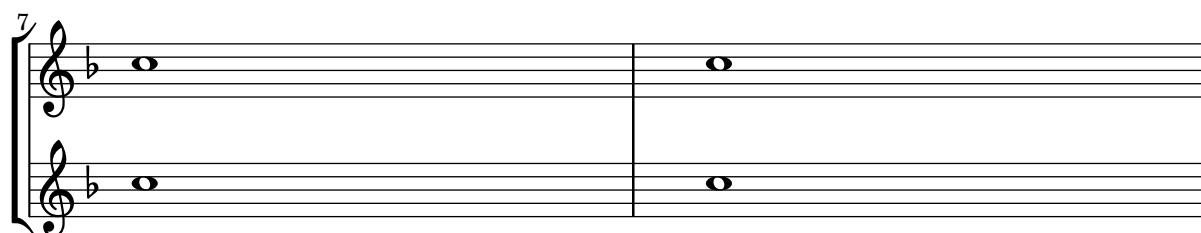
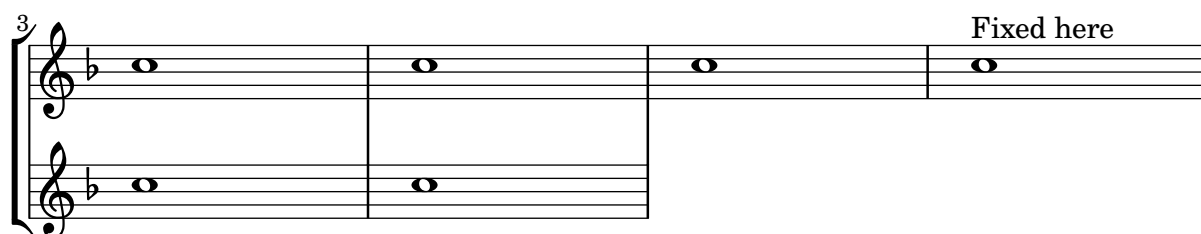
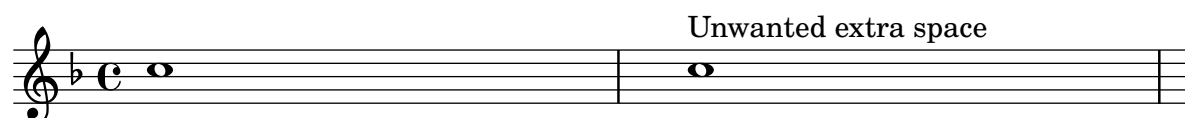
When adding a new staff at a line break, some extra space is unfortunately added at the end of the line before the break (to fit in a key signature change, which will never be printed anyway). The workaround is to add a setting of `Staff.explicitKeySignatureVisibility` as is shown in the example.

```
\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 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
}
}
>>
}

```



Automatically changing the stem direction of the middle note based on the melody

LilyPond can alter the stem direction of the middle note on a staff so that it follows the melody, by adding the `Melody_engraver` to the `Voice` context and overriding the `neutral-direction` of `Stem`.

```

\relative c'' {
  \time 3/4
  a8 b g f b g |
  c b d c b c |
}

\layout {
  \context {
    \Voice
    \consists "Melody_engraver"
    \autoBeamOff
    \override Stem.neutral-direction = #'()
  }
}

```



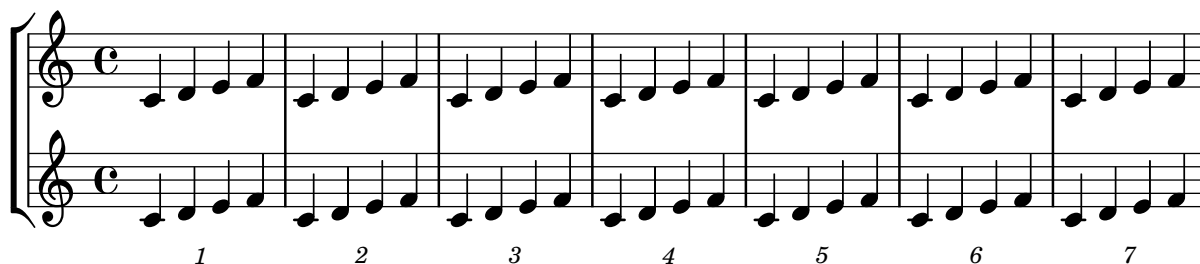
Centered measure numbers

Scores of large ensemble works often have bar numbers placed beneath the system, centered horizontally on the measure's extent. This snippet shows how the `Measure_counter_engraver` may be used to simulate this notational practice. Here, the engraver has been added to a `Dynamics` context.

```
\layout {
  \context {
    \Dynamics
    \consists #Measure_counter_engraver
    \override MeasureCounter.direction = #DOWN
    \override MeasureCounter.font-encoding = #'latin1
    \override MeasureCounter.font-shape = #'italic
    % to control the distance of the Dynamics context from the staff:
    \override VerticalAxisGroup.nonstaff-relatedstaff-spacing.padding = #2
  }
  \context {
    \Score
    \remove "Bar_number_engraver"
  }
}

pattern = \repeat unfold 7 { c'4 d' e' f' }

\new StaffGroup <<
  \new Staff {
    \pattern
  }
  \new Staff {
    \pattern
  }
  \new Dynamics {
    \startMeasureCount
    s1*7
    \stopMeasureCount
  }
>>
```



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`

The `measureLength` property, together with `measurePosition`, determines when a bar line is needed. However, when using `\scaleDurations`, the scaling of durations makes it difficult to change time signatures. In this case, `measureLength` should be set manually, using the `ly:make-moment` callback. The second argument must be the same as the second argument of `\scaleDurations`.

```

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

<<
  \new Staff {
    \scaleDurations 8/5 {
      \time 6/8
      \set Timing.measureLength = #(ly:make-moment 6/5)
      b8 b b b b b
      \time 2/4
      \set Timing.measureLength = #(ly:make-moment 4/5)
      b4 b
    }
  }
  \new Staff {
    \clef bass
    \time 2/4
    c2 d e f
  }
>>

```



Chant or psalms notation

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

```

stemOff = \hide Staff.Stem
stemOn  = \undo \stemOff

```

```

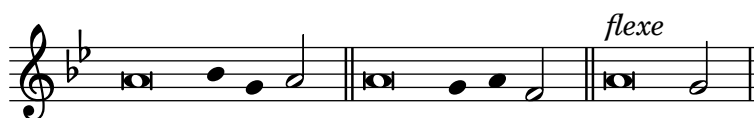
\score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \key g \minor
    \cadenzaOn
    \stemOff a'\breve bes'4 g'4
    \stemOn a'2 \bar "||"
    \stemOff a'\breve g'4 a'4
  }
}

```

```

\stemOn f'2 \bar "||"
\stemOff a'\breve^\markup { \italic flexe }
\stemOn g'2 \bar "||"
}
}

```



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 20)

\score {
  {
    \repeat unfold 12 { s1 \break }
  }
  \layout {
    indent = 0\in
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Clef_engraver"
      \remove "Bar_engraver"
    }
    \context {
      \Score
      \remove "Bar_number_engraver"
    }
  }
}

% 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
}
%}

% uncomment these lines for "A4" size
%{
\paper {
  #(set-paper-size "a4")
}
%}

```



```
ragged-last-bottom = ##f
line-width = 180
left-margin = 15
bottom-margin = 10
top-margin = 10
}
%}
```

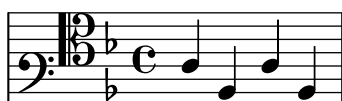
Creating custom key signatures

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

```
\new Staff \with {
  \override StaffSymbol.line-count = #8
  \override KeySignature.flat-positions = #'((-7 . 6))
  \override KeyCancellation.flat-positions = #'((-7 . 6))
  % presumably sharps are also printed in both octaves
  \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,
}
```



Cross staff stems

This snippet shows the use of the `Span_stem_engraver` and `\crossStaff` to connect stems across staves automatically.

The stem length need not be specified, as the variable distance between noteheads and staves is calculated automatically.

```
\layout {
  \context {
    \PianoStaff
    \consists #Span_stem_engraver
  }
}

{
  \new PianoStaff <<
    \new Staff {
      <b d'>4 r d'16\> e'8. g8 r\!
      e'8 f' g'4 e'2
    }
  }
}
```

```

\new Staff {
  \clef bass
  \voiceOne
  \autoBeamOff
  \crossStaff { <e g>4 e, g16 a8. c8} d
  \autoBeamOn
  g8 f g4 c2
}
>>
}

```



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 @var{grob-item} to the array of conditional elements of @var{grob}."
  Rewrite of @code{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 @var{accidental-grob}."
  Rewrite of @code{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 @var{accidental-grob}, an @code{Accidental} grob, to the
  list of the accidental grobs of @var{grob}, an @code{AccidentalPlacement}
  grob.
  Rewrite of @code{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 structure
%%%

%%% 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 <ambitus-note> 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 if 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)
  "If @var{direction} is @code{UP}, then return the upper ambitus note
of @var{ambitus}, otherwise return the lower ambitus note."
  (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 logics
 %%%
 %%% 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 contain 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 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 (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 the 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 @var{ambitus}, by getting the starting
position of middle C and key signature from @var{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 the upper and lower ambitus pithes of @var{ambitus}, using
@var{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,
          ;; and conversely for upper pitch.
          (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 DOWN UP))
    (list DOWN UP))

```

[illegible]

```

                                '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 notes ==> suicide the grobs
(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 orchestral scores sometimes single or groups of instruments are silent for a while and their staves can be removed for that time (with `\removeEmptyStaves`).

When they play again it is often preferred to show the staves of *all instruments of such a group*. This can be done adding the `Keep_alive_together_engraver` in the grouper (e.g., a `GrandStaff` or a `StaffGroup`).

In the example the violins are silent in the 2nd system and in the 3rd system. Only the first violin plays the last measure but the staff of the second violin is also displayed.

```

\score {
  <<
    \new StaffGroup = "StaffGroup_woodwinds"
    <<
      \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 \repeat unfold 7 { s1 } \repeat unfold 12 a16 a4
}
\new Staff = "StaffViolinII" \with {
  instrumentName = "Violin II"
  shortInstrumentName = "Vi II"
}
\relative c' { e1 \repeat unfold 8 { s1 } }
>>
\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 a musical score for four instruments: Flute, Violin I, Violin II, and Cello. The score is organized into two systems. In the first system, the Flute (Fl) plays a continuous eighth-note melody across three measures. Violin I (Vi I) and Violin II (Vi II) play sustained notes (half notes) in the first measure, while Violin II plays a whole note in the second measure. The Cello (Ce) plays a sustained note (half note) in the first measure, and a whole note in the second and third measures. The second system shows the Flute continuing its melody, while the Cello continues its sustained notes. The Violin staves are empty in the second system.

The image shows a musical score snippet with four staves. The first staff is labeled 'Fl' (Flute) and begins with a measure number '7'. It contains a sequence of eighth notes across three measures. The second and third staves are labeled 'Vi I' (Violin I) and 'Vi II' (Violin II) respectively, and are grouped with a brace on the left. The Violin I staff has a series of beamed sixteenth notes in the third measure. The fourth staff is labeled 'Ce' (Cello) and contains a single note in the third measure.

Engravers one-by-one

The notation problem, creating a certain symbol, is handled by plugins. Each plugin is called an Engraver. In this example, 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.

```
%% sample music
topVoice = \relative c' {
  \key d \major
  es8([ g] a[ fis])
  b4
  b16[-. b-. b-. cis-.]
  d4->
}
```

```
botVoice = \relative c' {
  \key d \major
  c8[( f] b[ a])
  es4
  es16[-. es-. es-. fis-.]
  b4->
}
```

```
hoom = \relative c {
  \key d \major
  \clef bass
  g8-. r
  r4
  fis8-.
  r8
  r4
}
```

```

    b'4->
}

pah = \relative c' {
    r8 b-.
    r4
    r8 g8-.
    r16 g-. r8
    \clef treble
    fis'4->
}

%
% setup for Request->Element conversion. Guru-only
%

MyStaff = \context {
    \type "Engraver_group"
    \name Staff

    \description "Handles clefs, bar lines, keys, accidentals. It can contain
@code{Voice} contexts."

    \consists "Output_property_engraver"

    \consists "Font_size_engraver"

    \consists "Volta_engraver"
    \consists "Separating_line_group_engraver"
    \consists "Dot_column_engraver"

    \consists "Ottava_spanner_engraver"
    \consists "Rest_collision_engraver"
    \consists "Piano_pedal_engraver"
    \consists "Piano_pedal_align_engraver"
    \consists "Instrument_name_engraver"
    \consists "Grob_pq_engraver"
    \consists "Forbid_line_break_engraver"
    \consists "Axis_group_engraver"

    \consists "Pitch_squash_engraver"

    localAlterations = #'()

    % explicitly set instrumentName, so we don't get
    % weird effects when doing instrument names for
    % piano staves

    instrumentName = #'()
    shortInstrumentName = #'()

    \accepts "Voice"

```

```
\defaultchild "Voice"
}
```

```
MyVoice = \context {
  \type "Engraver_group"
  \name Voice

  \description "
    Corresponds to a voice on a staff. This context handles the
    conversion of dynamic signs, stems, beams, super- and subscripts,
    slurs, ties, and rests.

    You have to instantiate this explicitly if you want to have
    multiple voices on the same staff."

  localAlterations = #'()
  \consists "Font_size_engraver"

  % must come before all
  \consists "Output_property_engraver"
  \consists "Arpeggio_engraver"
  \consists "Multi_measure_rest_engraver"
  \consists "Text_spanner_engraver"
  \consists "Grob_pq_engraver"
  \consists "Note_head_line_engraver"
  \consists "Glissando_engraver"
  \consists "Ligature_bracket_engraver"
  \consists "Breathing_sign_engraver"
  % \consists "Rest_engraver"
  \consists "Grace_beam_engraver"
  \consists "New_fingering_engraver"
  \consists "Chord_tremolo_engraver"
  \consists "Percent_repeat_engraver"
  \consists "Slash_repeat_engraver"

  %{
    Must come before text_engraver, but after note_column engraver.
  %}
  \consists "Text_engraver"
  \consists "Dynamic_engraver"
  \consists "Dynamic_align_engraver"
  \consists "Fingering_engraver"

  \consists "Script_column_engraver"
  \consists "Rhythmic_column_engraver"
  \consists "Cluster_spanner_engraver"
  \consists "Tie_engraver"
  \consists "Tie_engraver"
  \consists "Tuplet_engraver"
  \consists "Note_heads_engraver"
  \consists "Rest_engraver"
```

```
}
```

```
\score {  
  \topVoice  
  \layout {  
    \context { \MyStaff }  
    \context { \MyVoice }  
  }  
}
```

```
MyStaff = \context {  
  \MyStaff  
  \consists "Staff_symbol_engraver"  
}
```

```
\score {  
  \topVoice  
  \layout {  
    \context { \MyStaff }  
    \context { \MyVoice }  
  }  
}
```

```
MyStaff = \context {  
  \MyStaff  
  \consists "Clef_engraver"  
  \remove "Pitch_squash_engraver"  
}
```

```
\score {  
  \topVoice  
  \layout {  
    \context { \MyStaff }  
    \context { \MyVoice }  
  }  
}
```

```
MyVoice = \context {  
  \MyVoice  
  \consists "Stem_engraver"  
}
```

```
\score {  
  \topVoice  
  \layout {  
    \context { \MyStaff }  
    \context { \MyVoice }  
  }  
}
```

```

MyVoice = \context {
  \MyVoice
  \consists "Beam_engraver"
}

\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

MyVoice = \context {
  \MyVoice
  \consists "Phrasing_slur_engraver"
  \consists "Slur_engraver"
  \consists "Script_engraver"
}

\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

MyStaff = \context {
  \MyStaff
  \consists "Bar_engraver"
  \consists "Time_signature_engraver"
}

\score {
  \topVoice
  \layout {
    \context { \MyStaff }
    \context { \MyVoice }
  }
}

MyStaff = \context {
  \MyStaff
  \consists "Accidental_engraver"
  \consists "Key_engraver"
}

\score {
  \topVoice
  \layout {
    \context { \MyStaff }

```

```

\context { \MyVoice }
}
}

```



Mensurstriche layout (bar lines between the staves)

The mensurstriche-layout where the bar lines do not show on the staves but between staves can be achieved with a `StaffGroup` instead of a `ChoirStaff`. The bar line on staves is blanked out using `\hide`.

```

global = {
  \hide Staff.BarLine
  s1 s
  % the final bar line is not interrupted
  \undo \hide Staff.BarLine
  \bar "|"
}

```

```

\new StaffGroup \relative c'' {
  <<

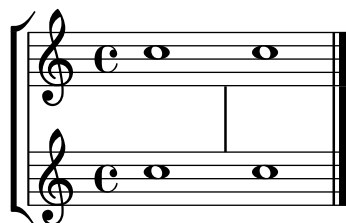
```



```

\new Staff { << \global { c1 c } >> }
\new Staff { << \global { c c } >> }
>>
}

```



Nesting staves

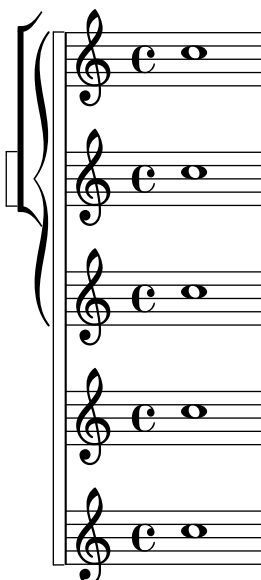
The property `systemStartDelimiterHierarchy` can be used to make more complex nested staff groups. The command `\set StaffGroup.systemStartDelimiterHierarchy` takes an alphabetical list of the number of staves produced. Before each staff a system start delimiter can be given. It has to be enclosed in brackets and takes as much staves as the brackets enclose. Elements in the list can be omitted, but the first bracket takes always the complete number of staves. The possibilities are `SystemStartBar`, `SystemStartBracket`, `SystemStartBrace`, and `SystemStartSquare`.

```

\new StaffGroup
\relative c'' <<
  \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 }
>>

```



Numbering groups of measures

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.

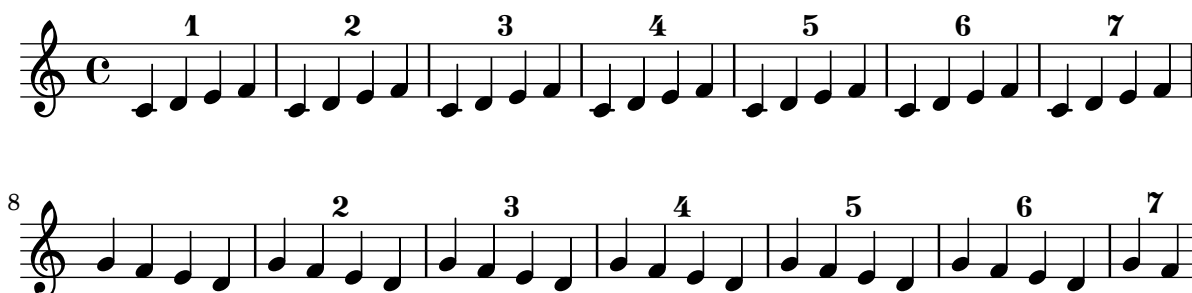
The engraver must be added to the appropriate context. Here, a `Staff` context is used; another possibility is a `Dynamics` context.

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
}
```



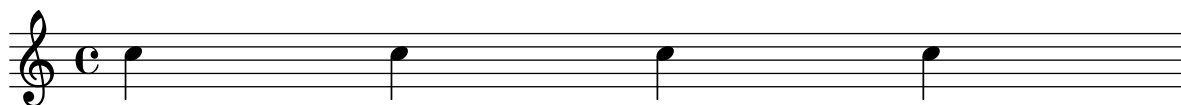


Removing bar numbers from a score

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
}
```



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 marklines in a Frenched score

Using `MarkLine` contexts (such as in LSR1010 (<http://lsr.di.unimi.it/LSR/Item?id=1010>)) 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
}

\layout {
  indent = 15\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
    \override VerticalAxisGroup.remove-empty = ##t
    \override VerticalAxisGroup.remove-layer = #'any
    \override VerticalAxisGroup.staff-affinity = #DOWN
    \override VerticalAxisGroup.nonstaff-relatedstaff-spacing.basic-distance = 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
}
}

\score {
  <<
    \new StaffGroup = "winds" \with {
      instrumentName = "Winds"
      shortInstrumentName = "Winds"
    } <<
      \new MarkLine \bars
      \new Staff \winds
    >>
    \new StaffGroup = "brass" <<
      \new MarkLine \bars
      \new Staff = "trumpet" \with {
        instrumentName = "Trumpet"
        shortInstrumentName = "Tpt"
      } \trumpet
      \new Staff = "trombone" \with {
        instrumentName = "Trombone"
        shortInstrumentName = "Tbn"
      } \trombone
    >>
    \new StaffGroup = "strings" \with {
      instrumentName = "Strings"
      shortInstrumentName = "Strings"
    } <<
      \new MarkLine \bars
      \new Staff = "strings" { \strings }
    >>
  >>
}

```

Allegro (♩ = 120)

Winds

Trumpet

Trombone

Strings

A

B

The musical score is divided into five systems, each with a 'Winds' and 'Strings' staff. The lyrics are represented by letters A through N. The tempo is marked 'Adagio' with a quarter note equal to 40 beats. The key signature is C major and the time signature is 4/4.

System 1 (Measures 6-10): Winds and Strings. Lyrics: C, D, C, D, C, D.

System 2 (Measures 11-15): Winds, Tbn, and Strings. Lyrics: E, F, E, F, E, F. Tempo: Adagio (♩ = 40).

System 3 (Measures 16-20): Winds and Strings. Lyrics: G, H, G, H, G, H.

System 4 (Measures 21-25): Winds, Tpt, and Strings. Lyrics: J, K, L, K, L, K.

System 5 (Measures 26-30): Winds and Strings. Lyrics: M, N, M, N, M, N.

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 which 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 |
}

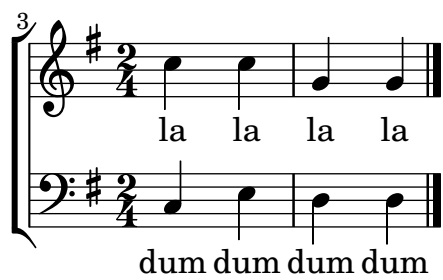
BassLyrics = \lyricmode {
  dum dum |
  dum dum |
}

\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
    }
  }
}

```



Tweaks and overrides

Section “Changing defaults” in *Notation Reference*

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

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



Adding links to objects

To add a link to a grob-stencil you could use `add-link` as defined here. Works with `\override` and `\tweak`.

Drawback: `point-and-click` will be 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)
          (begin
            (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)))
    #f)))

%%% test

urlI =
"https://lilypond.org/doc/v2.14/Documentation/notation/writing-pitches"

urlII =
"https://lilypond.org/doc/v2.14/Documentation/notation/rhythms"

urlIII =
"https://lilypond.org/doc/v2.14/Documentation/notation/note-heads"

urlIV =
"https://lilypond.org/doc/v2.14/Documentation/notation/beams"

urlV =
"https://lilypond.org/doc/v2.14/Documentation/notation/note-head-styles"

\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,
    % With 2.17.9 you could use the command below to address the Accidental.
    % \tweak Accidental.before-line-breaking #(add-link url)
    \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 does not show in a tablature.

To make them appear, simply use the command `\revert TabStaff.TextScript.stencil`

%% <http://lsr.di.unimi.it/LSR/Item?id=919>

% by P.P.Schneider on June 2014

```
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 = #(ly:make-moment 1/8)
    }
  }
}

```

	1	&	2	&	3	&	4	&	1	&	2	&	3	&	4	&
T	1	1							1	1						
A	0	0							0	0						
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 lyrics vertical spacing

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 }
>>
```



Altering the length of beamed stems

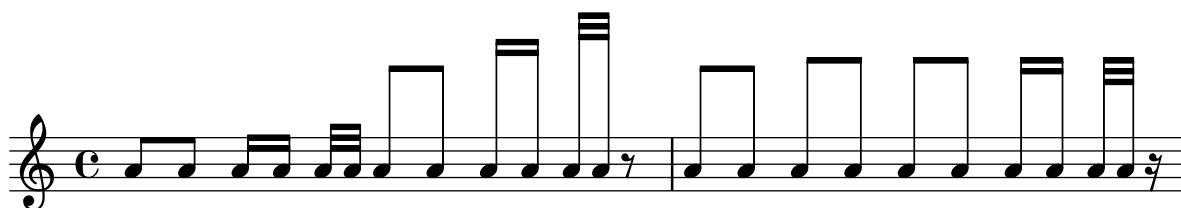
Stem lengths on beamed notes can be varied by overriding the `beamed-lengths` property of the `details` of the `Stem`. If a single value is used as an argument, the length applies to all stems. When multiple arguments are used, the first applies to eighth notes, the second to sixteenth notes and so on. 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
a8[ a] a16[ a] a32[ a] r16
}

```



Alternative bar numbering

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

```

\relative c' {
  \set Score.alternativeNumberingStyle = #'numbers
  \repeat volta 3 { c4 d e f | }
  \alternative {
    { c4 d e f | c2 d \break }
    { f4 g a b | f4 g a b | f2 a | \break }
    { c4 d e f | c2 d }
  }
  c1 \break
  \set Score.alternativeNumberingStyle = #'numbers-with-letters
  \repeat volta 3 { c,4 d e f | }
  \alternative {
    { c4 d e f | c2 d \break }
    { f4 g a b | f4 g a b | f2 a | \break }
    { c4 d e f | c2 d }
  }
  c1
}

```



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 may be added to analysis brackets through the `text` property of the `HorizontalBracketText` grob. Adding different texts to brackets beginning at the same time requires the `\tweak` command. Bracket text will be parenthesized after a line break.

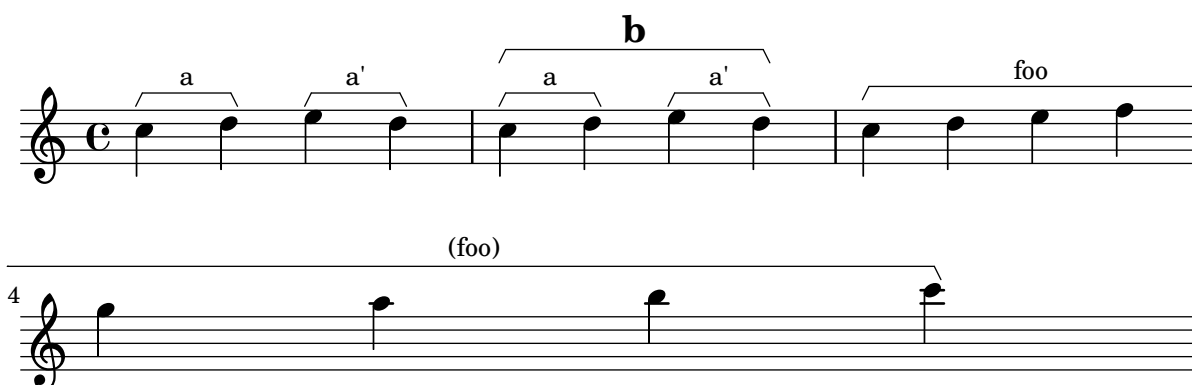
```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
    \override HorizontalBracket.direction = #UP
  }
}

{
  \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

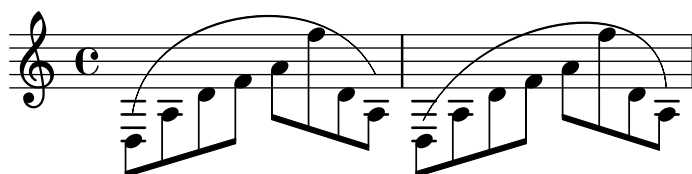
Slurs can be made asymmetric to match an asymmetric pattern of notes better.

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

```

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

```



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.

```

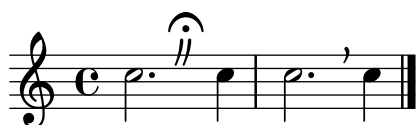
\relative c' {
  c2.
  % construct the symbol
  \override BreathingSign.text = \markup {
    \override #'(direction . 1)
    \override #'(baseline-skip . 1.8)
    \dir-column {

```

```

        \translate #'(0.155 . 0)
        \center-align \musicglyph "scripts.caesura.curved"
        \center-align \musicglyph "scripts.ufermata"
    }
}
\breathe c4
% set the breathe mark back to normal
\revert BreathingSign.text
c2. \breathe c4
\bar "|."
}

```



Changing a single note's size in a chord

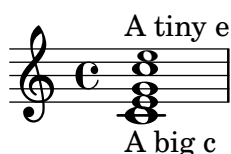
Individual note heads in a chord can be modified with the `\tweak` command inside a chord, by altering the `font-size` property.

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
}

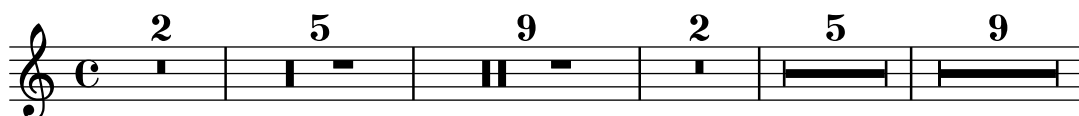
```



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 simple line is shown. This default number of ten may be changed by overriding the `expand-limit` property.

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



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.

```
#(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 with `make-pango-font-tree`.

```

\paper {
  % change for other default global staff size.
  myStaffSize = #20
  %{
    run
      lilypond -dshow-available-fonts
    to show all fonts available in the process log.
  %}

  #(define fonts
    (make-pango-font-tree "Times New Roman"
                          "Nimbus Sans,Nimbus Sans L"
                          "Luxi Mono"
                          "Helvetica"
                          "Courier"
                          (/ myStaffSize 20)))
}

\relative c' {
  c4^\markup {
    roman: foo \bold bla \italic bar \italic \bold baz
  }
  c'4_\markup {
    \override #'(font-family . sans)
    {
      sans: foo \bold bla \italic bar \italic \bold baz
    }
  }
  c'2^\markup {
    \override #'(font-family . typewriter)
    {
      mono: foo \bold bla \italic bar \italic \bold baz
    }
  }
}

```

mono: foo **bla** bar **baz**
 roman: foo **bla** *bar* **baz**



sans: foo **bla** *bar* **baz**

Changing the staff size

Though the simplest way to resize staves is to use `#{set-global-staff-size xx}`, 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 c
    }
  }
  \new Staff \with {
    fontSize = #-3
    \override StaffSymbol.staff-space = #(magstep -3)
  } {
    \clef bass
    c8 c c c c\f c c c
  }
>>
```



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

`Staff.pedalSustainStrings` can be used 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 used 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
}
```



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 { ragged-right = ##t }

\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 vertical ordering of scripts

The vertical ordering of scripts is controlled with the `'script-priority` property. The lower this number, the closer it will be put to the note. In this example, the `TextScript` (the *sharp* symbol) first has the lowest priority, so it is put lowest in the first example. In the second, the *prall trill* (the `Script`) has the lowest, so it is on the inside. When two objects have the same priority, the order in which they are entered determines which one comes first.

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

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



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.

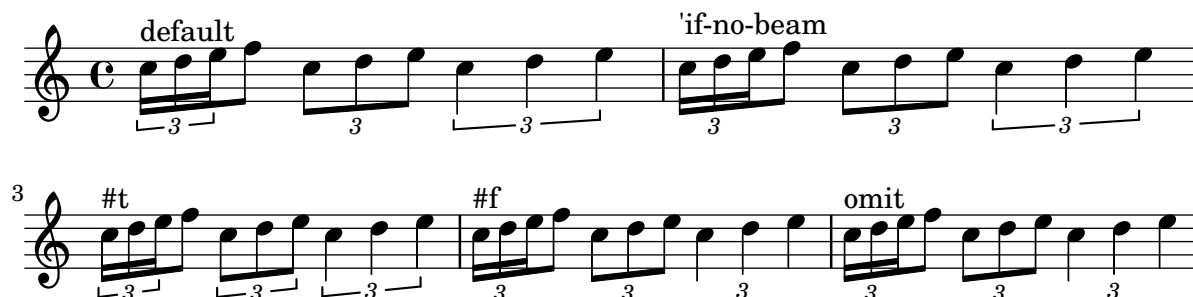
To control the visibility of tuplet brackets, set the property `'bracket-visibility` to either `#t` (always print a bracket), `'if-no-beam` (only print a bracket if there is no beam, which is the default behavior), or `#f` (never print a bracket). The latter is in fact equivalent to omitting the `@code{TupletBracket}` 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' {
    << \music s4^"default" >>
    \override TupletBracket.bracket-visibility = #'if-no-beam
    << \music s4^""if-no-beam" >>
    \override TupletBracket.bracket-visibility = ##t
    << \music s4^"#t" >>
    \override TupletBracket.bracket-visibility = ##f
    << \music s4^"#f" >>
    \omit TupletBracket
  }
}
```



```
<< \music s4~"omit" >>
}
```



Creating a delayed turn

Creating a delayed turn, where the lower note of the turn uses the accidental, requires several overrides. The `outside-staff-priority` property must be set to `#f`, as otherwise this would take precedence over the `avoid-slur` property. Changing the fractions `2/3` and `1/3` adjusts the horizontal position.

```
\relative c'' {
  c2*2/3 ( s2*1/3\turn d4) r
  <<
    { c4.( d8) }
    { s4 s\turn }
  >>
  \transpose c d \relative c'' <<
    { c4.( d8) }
    {
      s4
      \once \set suggestAccidentals = ##t
      \once \override AccidentalSuggestion.outside-staff-priority = ##f
      \once \override AccidentalSuggestion.avoid-slur = #'inside
      \once \override AccidentalSuggestion.font-size = -3
      \once \override AccidentalSuggestion.script-priority = -1
      \single \hideNotes
      b8-\turn \noBeam
      s8
    }
  >>
}
```



Creating custom key signatures

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

```
\new Staff \with {
  \override StaffSymbol.line-count = #8
```

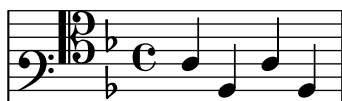
```

\override KeySignature.flat-positions = #'((-7 . 6))
\override KeyCancellation.flat-positions = #'((-7 . 6))
% presumably sharps are also printed in both octaves
\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,
}

```



Creating double-digit fingerings

Creating fingerings larger than 5 is possible.

```

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

```



Creating simultaneous rehearsal marks

Unlike text scripts, rehearsal marks cannot be stacked at a particular point in a score: only one `RehearsalMark` object is created. Using an invisible measure and bar line, an extra rehearsal mark can be added, giving the appearance of two marks in the same column.

This method may also prove useful for placing rehearsal marks at both the end of one system and the start of the following system.

```

{
  \key a \major
  \set Score.markFormatter = #format-mark-box-letters
  \once \override Score.RehearsalMark.outside-staff-priority = #5000
}

```

```

\once \override Score.RehearsalMark.self-alignment-X = #LEFT
\once \override Score.RehearsalMark.break-align-symbols = #'(key-signature)
\mark \markup { \bold { Senza denti } }

% the hidden measure and bar line
% \cadenzaOn turns off automatic calculation of bar numbers
\cadenzaOn
\once \override Score.TimeSignature.stencil = ##f
\time 1/16
s16 \bar ""
\cadenzaOff

\time 4/4
\once \override Score.RehearsalMark.self-alignment-X = #LEFT
\mark \markup { \box \bold Intro }
d'1
\mark \default
d'1
}

```



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

\set Staff.middleCPosition = #-13
\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 upper staff for creating cross-staff chords, because no problems with automatic beam collision avoidance then arise. If the stems from the lower staff were used in the following example, it would be necessary to change the automatic beam collision avoidance settings so that it doesn't detect collisions between staves using `\override Staff.Beam.collision-voice-only = ##t`

```

\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
}

```

>> }

```

\new Staff \with { \consists "Custos_engraver" } \relative c' {
  \override Staff.Custos.neutral-position = #4

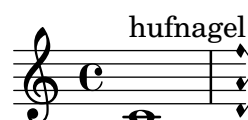
  \override Staff.Custos.style = #'hufnagel
  c1~"hufnagel" \break
  <d a' f'>1

  \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, therefore can be omitted in property overrides.

```

\include "predefined-guitar-fretboards.ly"
\storePredefinedDiagram #default-fret-table \chordmode { c' }
  #guitar-tuning
  #"x;1-1-(;3-2;3-3;3-4;1-1-);"

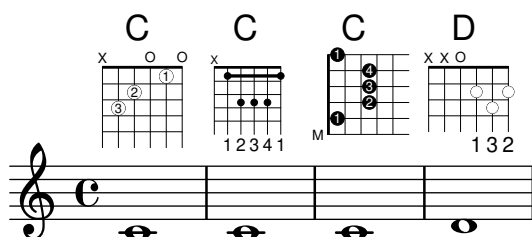
```

```

% shorthand
oo = #(define-music-function
  (grob-path value)
  (list? scheme?)
  #{ \once \override $grob-path = #value #})

<<
\new ChordNames {
  \chordmode { c1 | c | c | d }
}
\new FretBoards {
  % Set global properties of fret diagram
  \override FretBoards.FretBoard.size = #'1.2
  \override FretBoard.fret-diagram-details.finger-code = #'in-dot
  \override FretBoard.fret-diagram-details.dot-color = #'white
  \chordmode {
    c
    \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
  }
}
\new Voice {
  c'1 | c' | c' | d'
}
>>

```



Customizing markup fret diagrams

Fret diagram properties can be set through 'fret-diagram-details. For markup fret diagrams, overrides can be applied to the Voice.TextScript object or directly to the markup.

```
<<
\chords { c1 | c | c | d }

\new Voice = "mel" {
  \textLengthOn
  % Set global properties of fret diagram
  \override TextScript.size = #'1.2
  \override TextScript.fret-diagram-details.finger-code = #'in-dot
  \override TextScript.fret-diagram-details.dot-color = #'white

  %% C major for guitar, no barre, using defaults
  % terse style
  c'1^\markup { \fret-diagram-terse "x;3-3;2-2;o;1-1;o;" }

  %% C major for guitar, barred on third fret
  % verbose style
  % size 1.0
  % roman fret label, finger labels below string, straight barre
  c'1^\markup {
    % standard size
    \override #'(size . 1.0) {
      \override #'(fret-diagram-details . (
        (number-type . roman-lower)
        (finger-code . in-dot)
        (barre-type . straight))) {
        \fret-diagram-verbose #'(mute 6)
          (place-fret 5 3 1)
          (place-fret 4 5 2)
          (place-fret 3 5 3)
          (place-fret 2 5 4)
          (place-fret 1 3 1)
          (barre 5 1 3))
      }
    }
  }

  %% C major for guitar, barred on third fret
  % verbose style
  % landscape orientation, arabic numbers, M for mute string
  % no barre, fret label down or left, small mute label font
  c'1^\markup {
    \override #'(fret-diagram-details . (
      (finger-code . below-string)
      (number-type . arabic)
      (label-dir . -1)
      (mute-string . "M")
      (orientation . landscape)
      (barre-type . none)
```

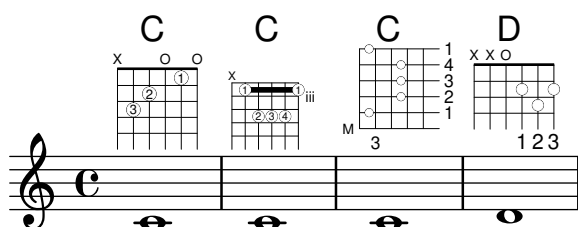


```

        (xo-font-magnification . 0.4)
        (xo-padding . 0.3))) {
  \fret-diagram-verbose #'((mute 6)
                           (place-fret 5 3 1)
                           (place-fret 4 5 2)
                           (place-fret 3 5 3)
                           (place-fret 2 5 4)
                           (place-fret 1 3 1)
                           (barre 5 1 3))
  }
}

%% simple D chord
% terse style
% larger dots, centered dots, fewer frets
% label below string
d'1^\markup {
  \override #'(fret-diagram-details . (
    (finger-code . below-string)
    (dot-radius . 0.35)
    (dot-position . 0.5)
    (fret-count . 3))) {
    \fret-diagram-terse "x;x;o;2-1;3-2;2-3;"
  }
}
}
>>

```



Display bracket with only one staff in a system

If there is only one staff in one of the staff types `ChoirStaff` or `StaffGroup`, by default the bracket and the starting bar line will not be displayed. This can be changed by overriding `collapse-height` to set its value to be less than the number of staff lines in the staff.

Note that in contexts such as `PianoStaff` and `GrandStaff` where the systems begin with a brace instead of a bracket, another property has to be set, as shown on the second system in the example.

```

\score {
  \new StaffGroup <<
    % Must be lower than the actual number of staff lines
    \override StaffGroup.SystemStartBracket.collapse-height = #4
    \override Score.SystemStartBar.collapse-height = #4
    \new Staff {
      c'1
    }
  }
}

```

```

>>
}
\score {
  \new PianoStaff <<
    \override PianoStaff.SystemStartBrace.collapse-height = #4
    \override Score.SystemStartBar.collapse-height = #4
    \new Staff {
      c'1
    }
  >>
}

```



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.

This macro prints (to the console) a textual representation of a grob's ancestry.

When called this way:

```
{ \once \override NoteHead.before-line-breaking = #display-ancestry c }
```

The following output is generated:

```

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
%% http://lsr.di.unimi.it/LSR/Item?id=622

```

%% see also <http://www.lilypond.org/doc/v2.18/Documentation/snippets/tweaks-and-overrides#t>

%% Remark:

%% grob::name is in the source since 2.19.x could be deleted during next LSR-upgrade

```

#(define (grob::name grob)
  (assq-ref (ly:grob-property grob 'meta) 'name))

#(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-error-port)
    "~3&~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.
}
```



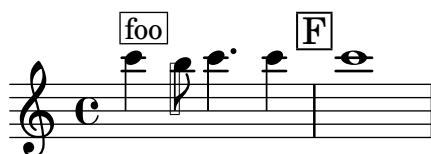
Drawing boxes around grobs

The `print`-function can be overridden to draw a box around an arbitrary grob.

```
\relative c' {
  \override TextScript.stencil =
    #(make-stencil-boxer 0.1 0.3 ly:text-interface::print)
  c'4~"foo"

  \override Stem.stencil =
    #(make-stencil-boxer 0.05 0.25 ly:stem::print)
  \override Score.RehearsalMark.stencil =
    #(make-stencil-boxer 0.15 0.3 ly:text-interface::print)
  b8

  \revert Stem.stencil
  \revert Flag.stencil
  c4. c4
  \mark "F"
  c1
}
```

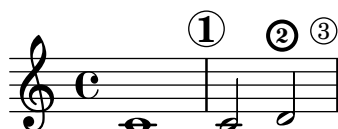


Drawing circles around various objects

The `\circle` markup command draws circles around various objects, for example fingering indications. For other objects, specific tweaks may be required: this example demonstrates two strategies for rehearsal marks and measure numbers.

```
\relative c' {
  c1
  \set Score.markFormatter =
    #(lambda (mark context)
      (make-circle-markup (format-mark-numbers mark context)))
  \mark \default

  c2 d^\markup {
    \override #'(thickness . 3) {
      \circle \finger 2
    }
  }
  \override Score.BarNumber.break-visibility = #all-visible
  \override Score.BarNumber.stencil =
    #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
}
```



Dynamics custom text spanner postfix

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

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

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

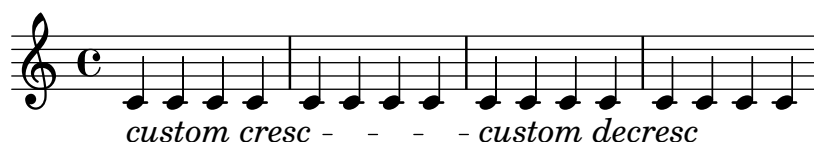
mydecresc =
  #(define-music-function (mymarkup) (markup?)
    (make-music 'DecrescendoEvent
      'span-direction START
      'span-type 'text
      'span-text mymarkup))
```

```
\relative c' {
  c4-\mycresc "custom cresc" c4 c4 c4 |
```

```

c4 c4 c4 c4 |
c4-\mydecrec "custom decrec" c4 c4 c4 |
c4 c4\! c4 c4
}

```



Dynamics text spanner postfix

Custom text spanners can be defined and used with hairpin and text crescendos. \< and \> produce hairpins by default, \cresc etc. produce text spanners by default.

% Some sample text dynamic spanners, to be used as postfix operators

```

crpoco =
#(make-music 'CrescendoEvent
      'span-direction START
      'span-type 'text
      'span-text "cresc. poco a poco")

```

```

\relative c' {
  c4\cresc d4 e4 f4 |
  g4 a4\! b4\crpoco c4 |
  c4 d4 e4 f4 |
  g4 a4\! b4\< c4 |
  g4\dim a4 b4\decrec c4\!
}

```



Extending a TrillSpanner

For TrillSpanner, the minimum-length property becomes effective only if the set-spacing-rods procedure is called explicitly.

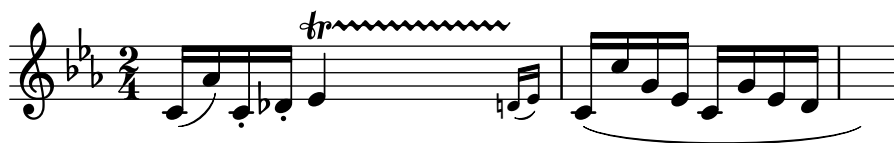
To do this, the springs-and-rods property should be set to ly:spanner::set-spacing-rods.

```

\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
\hideNotes
c8)

```

}



Extending glissandi across repeats

A glissando which extends into several `\alternative` blocks can be simulated by adding a hidden grace note with a glissando at the start of each `\alternative` block. The grace note should be at the same pitch as the note which starts the initial glissando. This is implemented here with a music function which takes the pitch of the grace note as its argument.

Note that in polyphonic music the grace note must be matched with corresponding grace notes in all other voices.

```
repeatGliss = #(define-music-function (grace)
  (ly:pitch?)
  #{
    % the next two lines ensure the glissando is long enough
    % to be visible
    \once \override Glissando.springs-and-rods
      = #ly:spanner::set-spacing-rods
    \once \override Glissando.minimum-length = #3.5
    \once \hideNotes
    \grace $grace \glissando
  })

\score {
  \relative c'' {
    \repeat volta 3 { c4 d e f\glissando }
    \alternative {
      { g2 d }
      { \repeatGliss f g2 e }
      { \repeatGliss f e2 d }
    }
  }
}

music = \relative c' {
  \voiceOne
  \repeat volta 2 {
    g a b c\glissando
  }
  \alternative {
    { d1 }
    { \repeatGliss c \once \omit StringNumber e1\2 }
  }
}

\score {
  \new StaffGroup <<
  \new Staff <<
```

```

\new Voice { \clef "G_8" \music }
>>
\new TabStaff <<
  \new TabVoice { \clef "moderntab" \music }
  >>
>>
}

```

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
  \once \override Staff.PianoPedalBracket.shorten-pair = #'(-7 . -2)
  c2\sostenutoOn c
  c2\sostenutoOff c
  \once \override Staff.PianoPedalBracket.edge-height = #'(0 . 3)
  c2\sostenutoOn c
  c2\sostenutoOff c
}

```

Flat Ties

The function takes the default `Tie.stencil` as an argument, calculating the result relying on the extents of this default.

Further tweaking is possible by overriding `Tie.details.height-limit` or with `\shape`. It's also possible to change the custom-definition on the fly.

%% <http://lsr.di.unimi.it/LSR/Item?id=1031>

```

\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)
      ;(coord-scale coord (cons x (* y dir)))
      (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))

;; outer let to trigger suicide
(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))))
      (uplist
        (map pair-to-list
          (normalize-coords coords lenx (* leny 2) dir))))

    (ly:stencil-translate
      (my-c-p-s uplist used-thick)
      (cons xtrans ytrans)))
  '()))

#(define flare-tie
  (flared-tie '((0 . 0)(0.1 . 0.2) (0.9 . 0.2) (1.0 . 0.0))))

\layout {
  \context {
    \Voice
    \override Tie.stencil = #flare-tie
  }
}

```

```

\paper { ragged-right = ##f }

\relative c' {
  a4~a
  \override Tie.height-limit = 4
  a'4~a
  a'4~a
  <a,, c e a c e a c e>~ q

  \break

  a'4~a
  \once \override Tie.details.height-limit = 14
  a4~a

  \break

  a4~a
  \once \override Tie.details.height-limit = 0.5
  a4~a

  \break

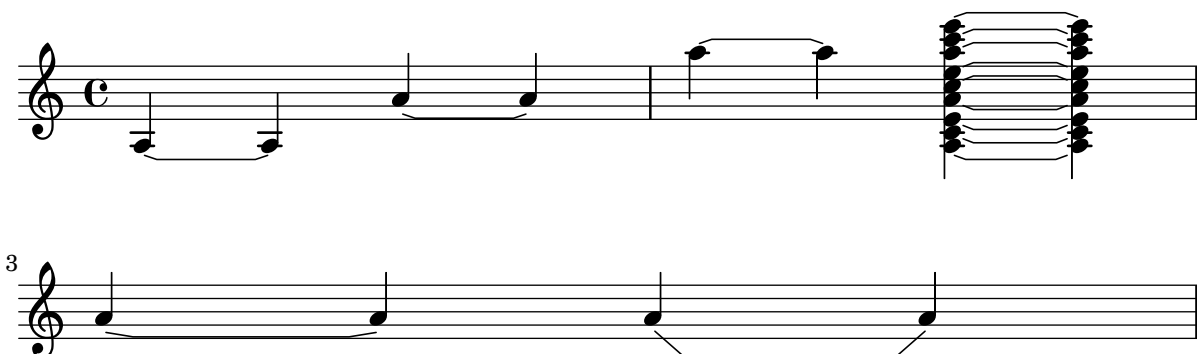
  a4~a
  \shape #'((0 . 0) (0 . 0.4) (0 . 0.4) (0 . 0)) Tie
  a4~a

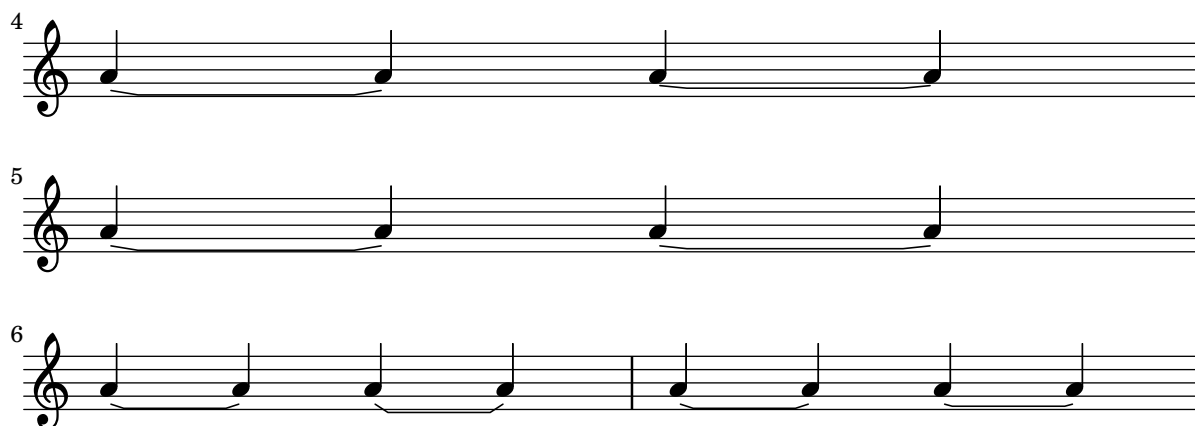
  \break

  a4~a
  \once \override Tie.stencil =
    #(flared-tie '((0 . 0)(0.1 . 0.4) (0.9 . 0.4) (1.0 . 0.0)))
  a4~a

  a4~a
  \once \override Tie.stencil =
    #(flared-tie '((0 . 0)(0.06 . 0.1) (0.94 . 0.1) (1.0 . 0.0)))
  a4~a
}

```





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 {
  a2 a
```

```

\repeat unfold 3 {
  c c c d d
}
}

\new Voice = "mel" {
  \textLengthOn
  % 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

  %% A chord for ukulele
  a'2^\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;"
    }
  }
}

%% 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'2^\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

%% C major for guitar, barred on third fret
% verbose style
% roman fret label, finger labels below string, straight barre
c'2^\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))
    }
  }
}

```

```

    }
  }
}

%% C major for guitar, barred on third fret
%% Double barre used to test barre function
% verbose style
c'2^\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))
      )
    }
  }
}

%% C major for guitar, with capo on third fret
% verbose style
c'2^\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))
      )
    }
  }
}

```

```

    }
}

%% simple D chord
d'2^\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;"
  }
}

%% simple D chord, large top fret thickness
d'2^\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

%% C major for guitar, barred on third fret
% verbose style
% roman fret label, finger labels below string, straight barre
c'2^\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))
    }
  }
}

%% C major for guitar, barred on third fret

```

```

%% Double barre used to test barre function
% verbose style
c'2^\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))
      }
    }
  }
}

%% C major for guitar, with capo on third fret
% verbose style
c'2^\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))
      }
    }
  }
}

%% simple D chord
d'2^\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;"
}
}

%% simple D chord, large top fret thickness
d'2^\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

%% C major for guitar, barred on third fret
% verbose style
% roman fret label, finger labels below string, straight barre
c'2^\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))
    }
  }
}

%% C major for guitar, barred on third fret
%% Double barre used to test barre function
% verbose style
c'2^\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))
    }
}
}

%% C major for guitar, with capo on third fret
% verbose style
c'2^\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))
    }
  }
}

%% simple D chord
d'2^\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;"
  }
}

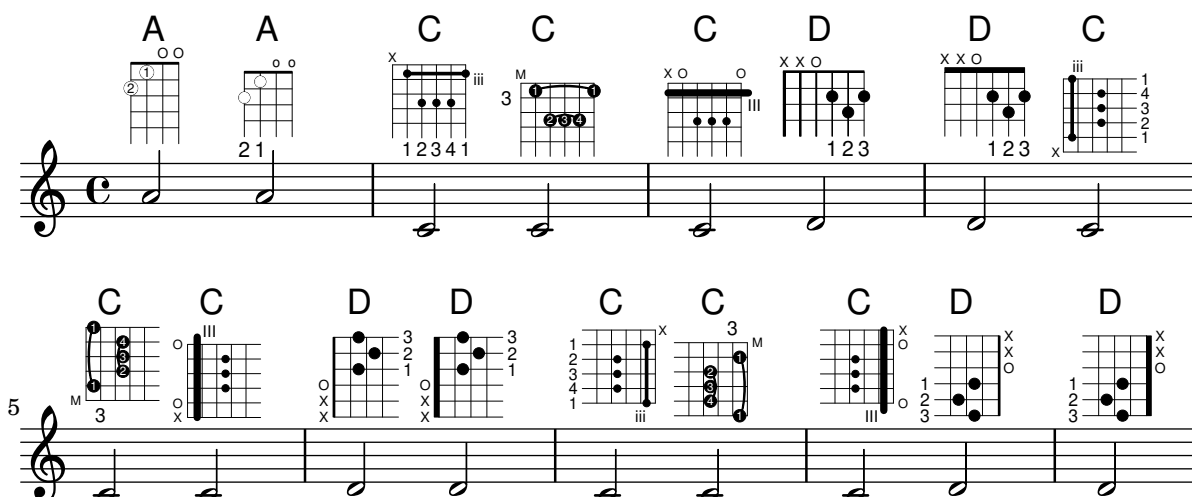
```

```

    }
  }

  %% simple D chord, large top fret thickness
  d'2^\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;"
    }
  }
}
>>

```



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
}

{
  \override Score.RehearsalMark.self-alignment-X = #LEFT
  \time 1/4
  \mark "Normal flags"
  \snippetexamplenotes

  \mark "Custom flag: inverted"
  \override Flag.stencil = #inverted-flag
  \snippetexamplenotes

  \mark "Custom flag: weight"
  \override Flag.stencil = #weight-flag
  \snippetexamplenotes

  \mark "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,
}

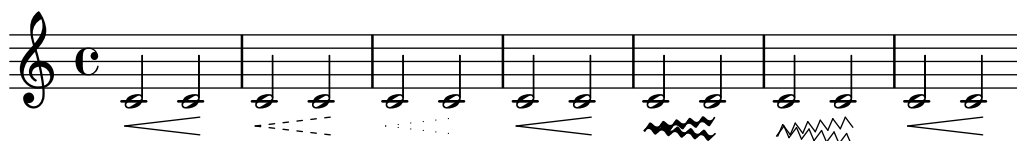
```



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 (e.g. "sempre pp" "piu f" "subito p")

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\overrideX-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: To LilyPond “sempre” has no extent, so it might put other stuff there and create collisions (which are not detected by the collision detection!). Also, there seems to be some spacing, so it’s not exactly the same alignment as without the additional text

- * Add an explicit shifting 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 = 2.5\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
sempMII =
#(make-dynamic-script
  (markup #:line (#:right-align
                  #:normal-text
                  #:italic "sempre"
                  #:dynamic "pp"))))

\new StaffGroup <<
  \new Staff = "s" \with { instrumentName = \markup \column { Normal } }
  <<
    \relative c'' {
      \key es \major
      c4\pp c\p c c | c\ff c c\pp c
    }
  >>
  \new Staff = "sMarkup" \with {
    instrumentName = \markup \column { Normal markup }
  }
  <<
    \relative c'' {
      \key es \major
      c4-\sempMMarkup c\p c c | c\ff c c-\sempMMarkup c
    }
  >>
  \new Staff = "sK" \with {
    instrumentName = \markup \column { Explicit shifting }
  }
  <<
    \relative c'' {
      \key es \major
      \once \override DynamicText.X-offset = #-9.2
      c4\sempMK c\p c c
      c4\ff c
      \once \override DynamicText.X-offset = #-9.2
      c4\sempMK c
    }
  >>
  \new Staff = "sT" \with {
    instrumentName = \markup \column { Right padding }
  }
  <<
    \relative c'' {

```

```

        \key es \major
        c4\semppT c\p c c | c\ff c c\semppT c
    }
>>
\new Staff = "sM" \with {
    instrumentName = \markup \column { Set dimension "to zero" }
}
<<
    \relative c'' {
        \key es \major
        c4\semppM c\p c c | c\ff c c\semppM c
    }
>>
\new Staff = "sG" \with {
    instrumentName = \markup \column { Shift inside dynamics}
}
<<
    \relative c'' {
        \key es \major
        c4\semppG c\p c c | c\ff c c\semppG c
    }
>>
\new Staff = "sMII" \with {
    instrumentName = \markup \column { Alignment inside dynamics }
}
<<
    \relative c'' {
        \key es \major
        % Setting to ##f (false) gives the same result
        \override DynamicText.X-offset = #0
        c4\semppMII c\p c c | c\ff c c\semppMII c
    }
>>
>>

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

```

Normal	
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 #'padding or #'extra-offset values (as shown in the first snippet)
- 2) you can add an invisible voice and attach the fret diagrams to the invisible notes in that voice (as shown in the second example).

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
  % THE FOLLOWING IS THE COMMAND TO MOVE THE CHORD NAME
  \once \override ChordNames.ChordName.extra-offset = #'(10 . 0)
  b8:13 s2.
  % THIS LINE IS THE SECOND METHOD
  s4 s4 b4:13
}

\score
{
  <<
    \new ChordNames \harmonies
    \new Staff
    {a8~\markup { \fret-diagram "6-x;5-0;4-2;3-0;2-0;1-2;" }
  % THE FOLLOWING IS THE COMMAND TO MOVE THE FRET DIAGRAM
  \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

```



```
% HERE IS THE SECOND METHOD
```

```
<<
  { a8 b4.~ b4. a8}
  { s4 s4 s4^ \markup { \fret-diagram "6-x;5-2;4-4;3-2;2-2;1-4;" }
  }
>>
}
>>
}
```

The image shows two musical staves. The first staff has two rehearsal marks: 'A 9 13' and 'B 9 13', both positioned above the staff. The second staff has one rehearsal mark: 'B 9 13', also positioned above the staff. Both staves show a melody with a slur over two notes.

How to print two rehearsal marks above and below the same barline (method 1)

This method prints two 'rehearsal marks', one on top of the other. It shifts the lower rehearsal mark below the staff and then adds padding above it in order to place the upper rehearsal mark above the staff.

By adjusting the extra-offset and baseline-skip values you can increase or decrease the overall space between the rehearsal mark and the staff.

Because nearly every type of glyph or string can be made to behave like a rehearsal mark it is possible to centre those above and below a bar line.

Adding the appropriate 'break visibility' as shown in snippet 1 (<http://lsr.di.unimi.it/LSR/Item?id=1>) will allow you to position two marks at the end of a line as well.

Note: Method 1 is less complex than Method 2 but does not really allow for fine tuning of placement of one of the rehearsal marks without affecting the other. It may also give some problems with vertical spacing, since using `extra-offset` does not change the bounding box of the mark from its original value.

```
\relative c'{
  c d e f |
  \once \override Score.RehearsalMark.extra-offset = #'(0 . -8.5)
  \once \override Score.RehearsalMark.baseline-skip = #9
  \mark \markup \center-column { \circle 1 \box A }
  g f e d |
  \once \override Score.RehearsalMark.extra-offset = #'(0 . -8.5)
  \once \override Score.RehearsalMark.baseline-skip = #9
  \mark \markup \center-column { \flat { \bold \small \italic Fine. } }
  g f e d |
```

```

\once \override Score.RehearsalMark.extra-offset = #'(0 . -8.5)
\once \override Score.RehearsalMark.baseline-skip = #9
\override Score.RehearsalMark.break-visibility = #begin-of-line-invisible
\mark \markup \center-column { \fermata \box z }
}

```



How to print two rehearsal marks above and below the same barline (method 2)

This method prints two 'rehearsal marks' - one above the staff and one below, by creating two voices, adding the Rehearsal Mark engraver to each voice - without this no rehearsal mark is printed - and then placing each rehearsal mark UP and DOWN in each voice respectively.

This method (as opposed to method 1) is more complex, but allows for more flexibility, should it be needed to tweak each rehearsal mark independently of the other.

```

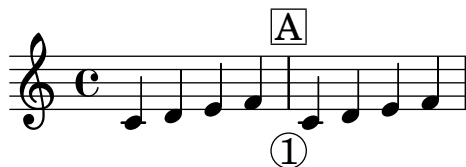
\score {
  \relative c'
  <<
    \new Staff {
      <<
        \new Voice \with {
          \consists Mark_engraver
          \consists "Staff_collecting_engraver"
        }
        { c4 d e f
          \mark \markup { \box A }
          c4 d e f
        }
        \new Voice \with {
          \consists Mark_engraver
          \consists "Staff_collecting_engraver"
          \override RehearsalMark.direction = #DOWN
        }
        { s4 s s s
          \mark \markup { \circle 1 }
          s4 s s s
        }
      >>
    }
  >>
  \layout {
    \context {
      \Score
      \remove "Mark_engraver"
      \remove "Staff_collecting_engraver"
    }
  }
}

```

```

    }
  }
}

```



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
}

```



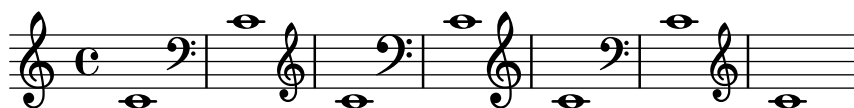
Keep change clefs full sized

When a clef is changed, the clef sign displayed is smaller than the initial clef. This can be overridden with `full-size-change`.

```

\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 the Glissando).

```
\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 with 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
  }
}
```

```
>>
}
```

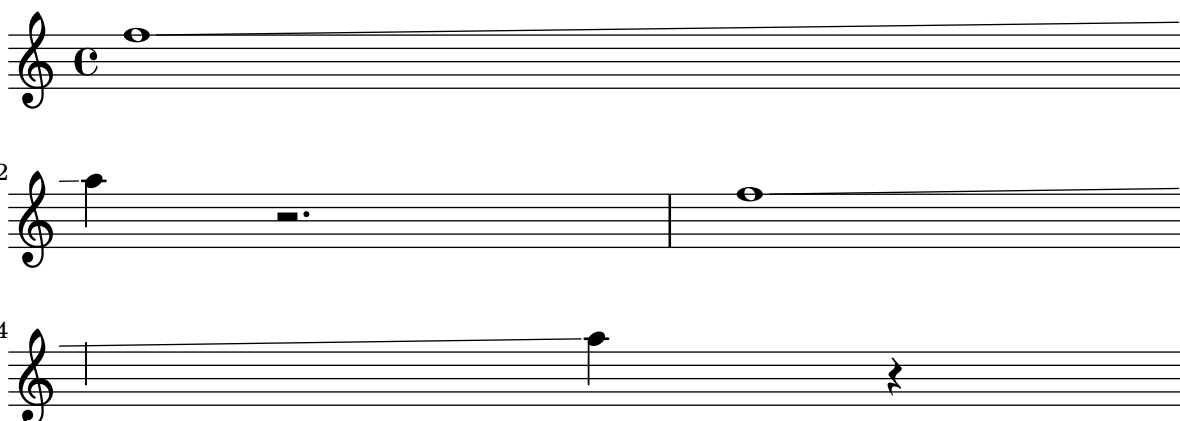


Making glissandi breakable

Setting the `breakable` property to `#t` in combination with `after-line-breaking` allows a glissando to break if it occurs at a line break:

```
glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}

\relative c' {
  \override Glissando.breakable = ##t
  \override Glissando.after-line-breaking = ##t
  f1\glissando |
  \break
  a4 r2. |
  f1\glissando
  \once \glissandoSkipOn
  \break
  a2 a4 r4 |
}
```



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
}
```

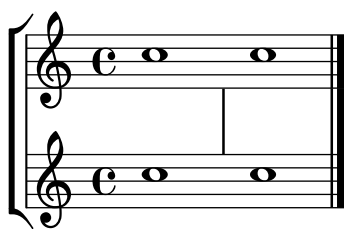


Mensurstriche layout (bar lines between the staves)

The mensurstriche-layout where the bar lines do not show on the staves but between staves can be achieved with a `StaffGroup` instead of a `ChoirStaff`. The bar line on staves is blanked out using `\hide`.

```
global = {
  \hide Staff.BarLine
  s1 s
  % the final bar line is not interrupted
  \undo \hide Staff.BarLine
  \bar "|"
}

\new StaffGroup \relative c'' {
  <<
    \new Staff { << \global { c1 c } >> }
    \new Staff { << \global { c c } >> }
  >>
}
```



Modifying the Ottava spanner slope

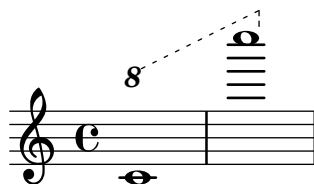
It is possible to change the slope of the Ottava spanner.

```
\relative c'' {
  \override Staff.OttavaBracket.stencil = #ly:line-spanner::print
  \override Staff.OttavaBracket.bound-details =
    #`((left . ((Y . 0) ; Change the integer here
      (attach-dir . ,LEFT)
      (padding . 0)
      (stencil-align-dir-y . ,CENTER)))
    (right . ((Y . 5) ; Change the integer here
      (padding . 0)
      (attach-dir . ,RIGHT)
      (text . ,(make-draw-dashed-line-markup
        (cons 0 -1.2))))))
```

```

\override Staff.OttavaBracket.left-bound-info =
  #ly:line-spanner::calc-left-bound-info-and-text
\override Staff.OttavaBracket.right-bound-info =
  #ly:line-spanner::calc-right-bound-info
\ottava #1
c1
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.

```

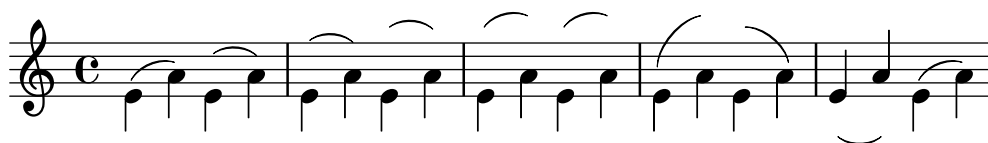
\relative c' {
  \stemDown
  e4( a)
  \override Slur.positions = #'(1 . 1)
  e4( a)
  \override Slur.positions = #'(2 . 2)
  e4( a)
}

```

```

\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)
}

```



Nesting staves

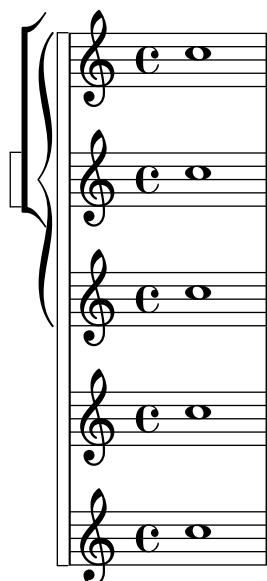
The property `systemStartDelimiterHierarchy` can be used to make more complex nested staff groups. The command `\set StaffGroup.systemStartDelimiterHierarchy` takes an alphabetical list of the number of staves produced. Before each staff a system start delimiter can be given. It has to be enclosed in brackets and takes as much staves as the brackets enclose. Elements in the list can be omitted, but the first bracket takes always the complete number of staves. The possibilities are `SystemStartBar`, `SystemStartBracket`, `SystemStartBrace`, and `SystemStartSquare`.

```

\new StaffGroup
\relative c'' <<
  \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 of distinct 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.

With 2.16.2 it is possible to put the proposed function, `\customScripts`, into a `\layout-block`.

% Code by David Nalesnik and Thomas Morley

```
#(define (custom-script-tweaks ls)
  (lambda (grob)
    (let* ((type (ly:prob-property
                  (ly:grob-property grob 'cause)
                  'articulation-type))
           (tweaks (assoc-ref ls type)))
      (if 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 a list of desired tweaks.
#(define my-settings-1
  '(
```

```

("staccato" . ((color . (1 0 0))(padding . 0.5)))
("accent" . ((font-size . 0)(color . (1 0 0))))
("tenuto" . ((rotation . (45 0 0)) (padding . 2)(font-size . 10)))
("staccatissimo" . ((padding . 1) (color . (1 0 0))))
("segno" . ((font-size . 0)(color . (1 0 0))))
))

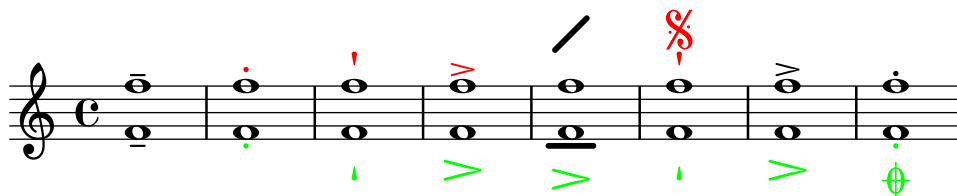
#(define my-settings-2
' (
  ("staccato" . ((color . (0 1 0))))
  ("accent" . ((font-size . 4)(color . (0 1 0))(padding . 1.5)))
  ("tenuto" . ((font-size . 10)))
  ("staccatissimo" . ((padding . 2) (color . (0 1 0))))
  ("coda" . ((color . (0 1 0)) (padding . 1)))
))

one =
\relative c' {
  f1--
  \customScripts #my-settings-1
  f-. f-! f-> f-- f-!\segno
  \revertCustomScripts
  f-> f-.
}

two =
\relative c' {
  f1--
  \customScripts #my-settings-2
  f-. f-! f-> f---> f-!
  f-> f-.\coda
}

\new Staff <<
  \new Voice { \voiceOne \one }
  \new Voice { \voiceTwo \two }
  >>

```



Percent repeat count visibility

Percent repeat counters can be shown at regular intervals by setting the context property `repeatCountVisibility`.

```

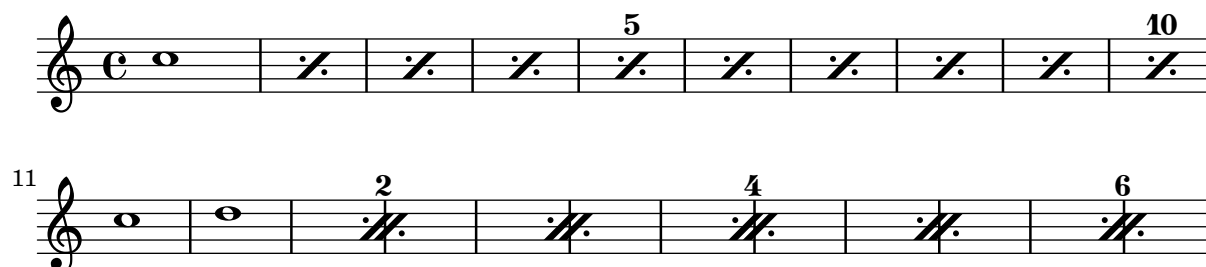
\relative c' {
  \set countPercentRepeats = ##t
  \set repeatCountVisibility = #(every-nth-repeat-count-visible 5)

```

```

\repeat percent 10 { c1 } \break
\set repeatCountVisibility = #(every-nth-repeat-count-visible 2)
\repeat percent 6 { c1 d1 }
}

```



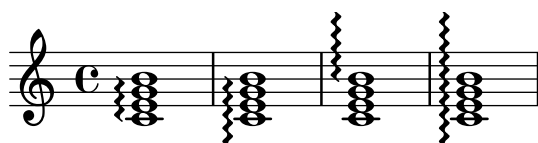
Positioning arpeggios

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 multi-measure rests

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

```

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

```

```

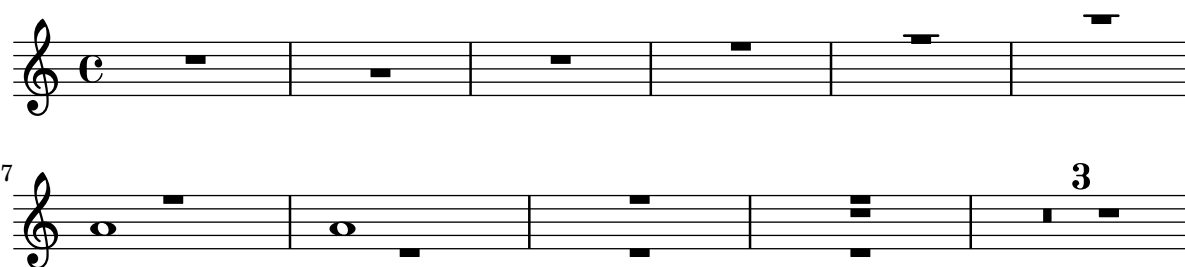
R1
\revert MultiMeasureRest.staff-position
\break

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

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

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

```



Positioning text markups inside slurs

Text markups need to have the `outside-staff-priority` property set to false in order to be printed inside slurs.

```

\relative c'' {
  \override TextScript.avoid-slur = #'inside
  \override TextScript.outside-staff-priority = ##f
  c2(~\markup { \halign #-10 \natural } d4.) c8
}

```



Printing bar numbers inside boxes or circles

Bar numbers can also be printed inside boxes or circles.

```
\relative c' {
  % Prevent bar numbers at the end of a line and permit them elsewhere
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 4)

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

  % Draw a box round the following bar number(s)
  \override Score.BarNumber.stencil
    = #(make-stencil-boxer 0.1 0.25 ly:text-interface::print)
  \repeat unfold 5 { c1 }

  % Draw a circle round the following bar number(s)
  \override Score.BarNumber.stencil
    = #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
  \repeat unfold 4 { c1 } \bar "|."
}
```



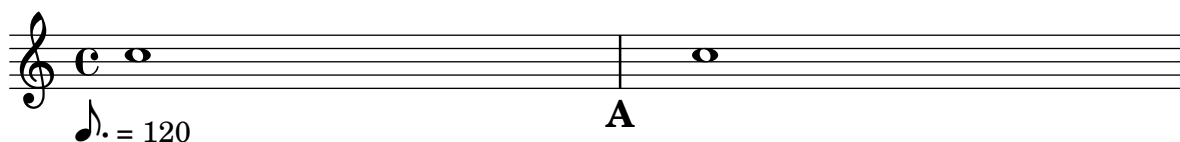
Printing metronome and rehearsal marks below the staff

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

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

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

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



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
    }
  >>
  R1
  <<
    \scale
    \context NoteNames {
      \set printOctaveNames = ##t
    }
  >>
}
```



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 = #(ly:make-moment 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

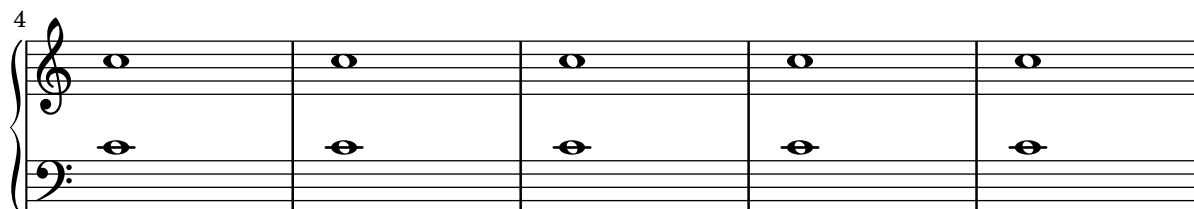
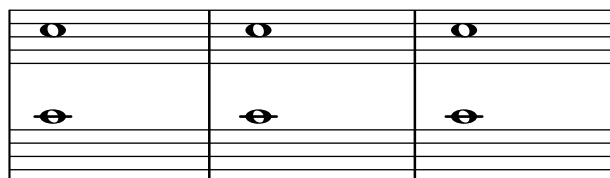
This snippet removes the first brace from a `PianoStaff` or a `GrandStaff`.

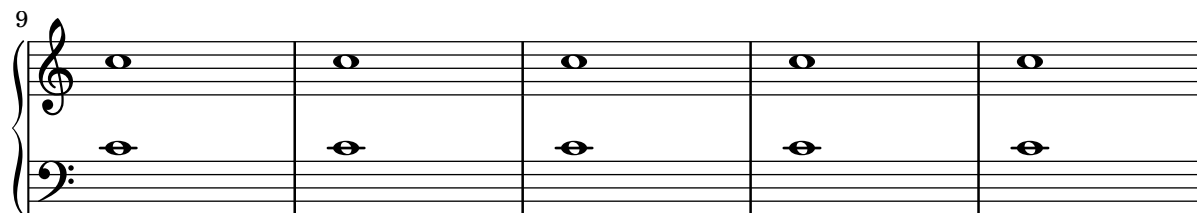
It may be useful when cutting and pasting the engraved image into existing music.

It uses `\alterBroken`.

```
someMusic = {
  \once \override Staff.Clef.stencil = ##f
  \once \override Staff.TimeSignature.stencil = ##f
  \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
    \context {
      \PianoStaff
      \alterBroken transparent #'(#t) SystemStartBrace
    }
  }
}
```

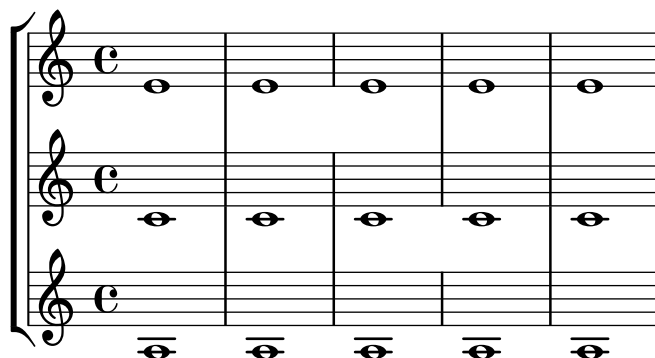




Removing connecting bar lines on StaffGroup, PianoStaff, or GrandStaff

By default, bar lines in StaffGroup, PianoStaff, or GrandStaff groups are connected between the staves, i.e. a SpanBar 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

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

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

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

```

Rest styles

Rests may be used in various styles.

```

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

```

```
\cadenzaOn
```

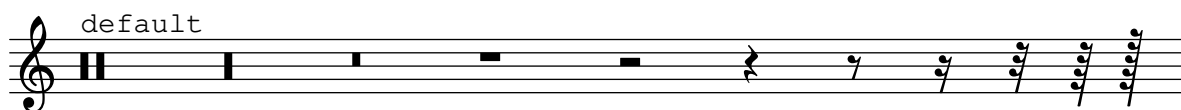
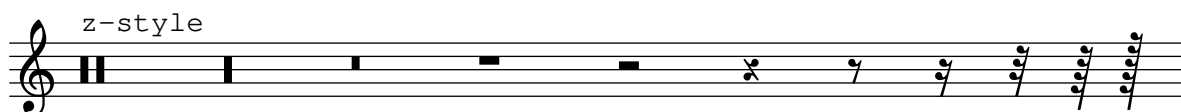
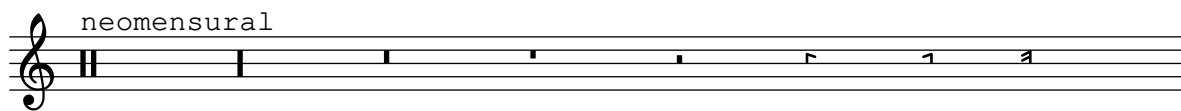
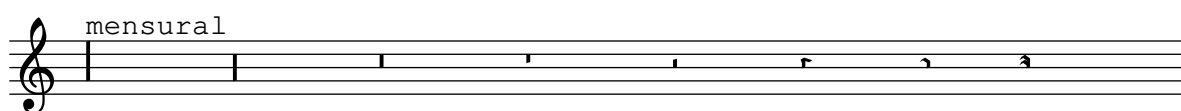
```
\override Staff.Rest.style = #'mensural
r\maxima^\markup \typewriter { mensural }
r\longa r\breve r1 r2 r4 r8 r16 s32 s64 s128 s128
\bar ""
\break
```

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

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

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

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



Rhythmic slashes

In “simple” lead-sheets, sometimes no actual notes are written, instead only “rhythmic patterns” and chords above the measures are notated giving the structure of a song. Such a feature is for example useful while creating/transcribing the structure of a song and also when sharing lead sheets with guitarists or jazz musicians.

The standard support for this using `\repeat percent` is unsuitable here since the first beat has to be an ordinary note or rest.

This example shows two solutions to this problem, by redefining ordinary rests to be printed as slashes. (If the duration of each beat is not a quarter note, replace the `r4` in the definitions with a rest of the appropriate duration).

```
% Macro to print single slash
rs = {
  \once \override Rest.stencil = #ly:percent-repeat-item-interface::beat-slash
  \once \override Rest.thickness = #0.48
  \once \override Rest.slope = #1.7
  r4
}

% Function to print a specified number of slashes
comp = #(define-music-function (count) (integer?)
  #{
    \override Rest.stencil = #ly:percent-repeat-item-interface::beat-slash
    \override Rest.thickness = #0.48
    \override Rest.slope = #1.7
    \repeat unfold $count { r4 }
    \revert Rest.stencil
  }
)

\score {
  \relative c' {
    c4 d e f |
    \rs \rs \rs \rs |
    \comp #4 |
  }
}
```



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. This example only modifies the second list, moving `key-cancellation` before `staff-bar`; by modifying the second list, break alignment behavior only changes in the middle of a system, not at the beginning or the end.

```
\new Staff {
```

```

\override Score.BreakAlignment.break-align-orders =
  ##((left-edge ambitus breathing-sign clef staff-bar
      key-cancellation key-signature time-signature custos)

      (left-edge ambitus breathing-sign clef key-cancellation
        staff-bar key-signature time-signature custos)

      (left-edge ambitus breathing-sign clef key-cancellation
        key-signature staff-bar time-signature custos))

\key des \major
c'1
\bar "||"
\key bes \major
c'1
}

```



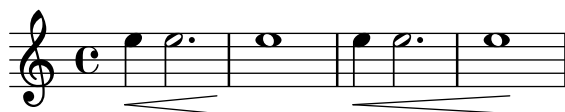
Setting hairpin behavior at bar lines

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 system separators

System separators can be inserted between systems. Any markup can be used, but `\slashSeparator` has been provided as a sensible default.

```

\paper {
  system-separator-markup = \slashSeparator
  line-width = 120
}

notes = \relative c' {
  c1 | c \break
  c1 | c \break
  c1 | c
}

```

```
\book {  
  \score {  
    \new GrandStaff <<  
      \new Staff \notes  
      \new Staff \notes  
    >>  
  }  
}
```

Three systems of musical notation, each consisting of a grand staff (treble and bass clef) and a common time signature 'C'. Each system contains two measures of music. The first system has a double bar line. The second system has a double bar line and a '3' below the first measure. The third system has a double bar line and a '5' below the first measure. Each measure contains a single note on the treble staff and a single note on the bass staff.

Showing the same articulation above and below a note or chord

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

So, the solution is to write the note as a chord and add the articulations inside the `<...>`. The direction will always be above, but one can tweak this via a `\tweak`: `<c-\tweak direction #DOWN-\fermata^\fermata>`

```
% The same as \flageolet, just a little smaller
smallFlageolet = \tweak font-size #-2 \flageolet
```

```
\relative c' {
  s4^"Wrong:"
  c4_\fermata^\fermata % The second fermata is ignored!
  <e d'>4^\smallFlageolet_\smallFlageolet

  s4^"Works if written inside a chord:"
  <e_\smallFlageolet d'^\smallFlageolet>4
  <e_\flageolet d'^\flageolet>4
  <e_\smallFlageolet^\smallFlageolet>4
  <e_\fermata^\fermata>4
}
```



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"
  \stringNumberSpanner "5"
```

```

\textSpannerDown
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**’ will appear 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 parentheses - method 3

Another way to put the time signature in parenthesis

```

\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 in parentheses

The time signature can be enclosed within parentheses.

```
\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 printing only the numerator as a number (instead of the fraction)

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

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



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
```

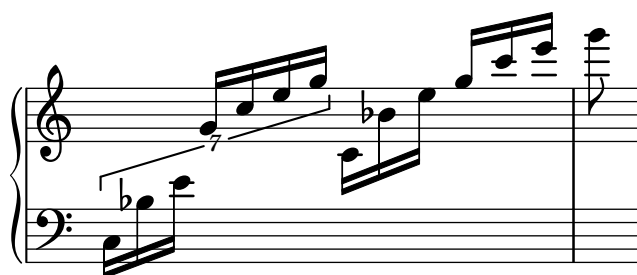
```

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 `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

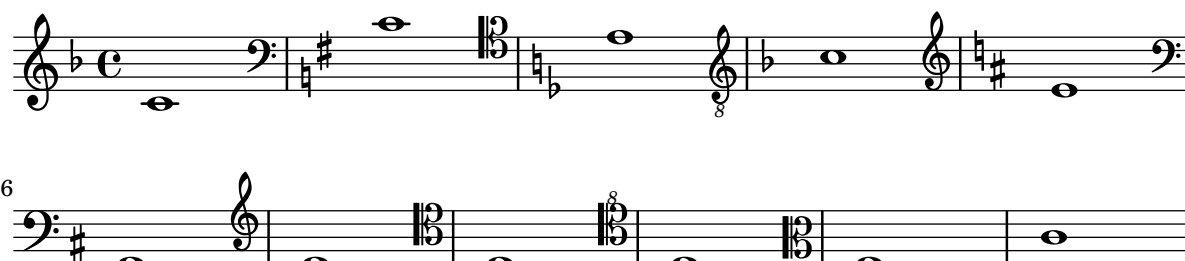
% The following clef changes do not preserve
% the normal relationship between notes, key signatures
% and clefs:

\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 layout of grace expressions can be changed throughout the music using the functions `add-grace-property` and `remove-grace-property`.

The following example undefines the `Stem` direction for this grace, so that stems do not always point up, and changes the default note heads to crosses.

```
\relative c' {
  \new Staff {
    $(remove-grace-property 'Voice 'Stem 'direction)
    $(add-grace-property 'Voice 'NoteHead 'style 'cross)
    \new Voice {
      \acciaccatura { f16 } g4
      \grace { d16 e } f4
      \appoggiatura { f,32 g a } e2
    }
  }
}
```



Using alternative flag styles

Alternative styles of flag on eighth and shorter notes can be displayed by overriding the `stencil` property of `Flag`. Valid values are `modern-straight-flag`, `old-straight-flag` and `flat-flag`.

```
testnotes = {
  \autoBeamOff
  c8 d16 c32 d64 \acciaccatura { c8 } d64 r4
}

\score {
  \relative c' {
    \time 2/4
    \testnotes

    \override Flag.stencil = #modern-straight-flag
```

```

\testnotes

\override Flag.stencil = #old-straight-flag
\testnotes

\override Flag.stencil = #flat-flag
\testnotes

\revert Flag.stencil
\testnotes
}
\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 example function defined below (“display-grobs”) is probably not that useful, but it demonstrates that the grobs are indeed being accessed.

Example console output:

```

#Grob Accidental () #Grob Stem
#(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 PostScript to generate special note head shapes

When a note head with a special shape cannot easily be generated with graphic markup, PostScript code can be used to generate the shape. This example shows how a parallelogram-shaped note head is generated.

```

parallelogram =
  #(ly:make-stencil (list 'embedded-ps
    "gsave
      currentpoint translate
      newpath
      0 0.25 moveto
      1.3125 0.75 lineto
      1.3125 -0.25 lineto
      0 -0.75 lineto
      closepath
      fill
      grestore" )
    (cons 0 1.3125)
    (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
}

```



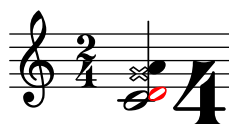
Using the `\tweak` command to tweak individual grobs

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.

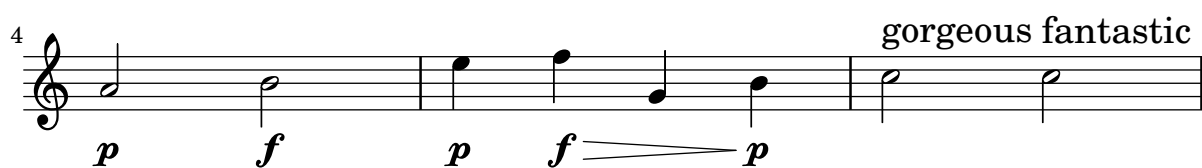
The same idea, together with `\textLengthOn`, is used to align the text scripts along their baseline.

```

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
}

```

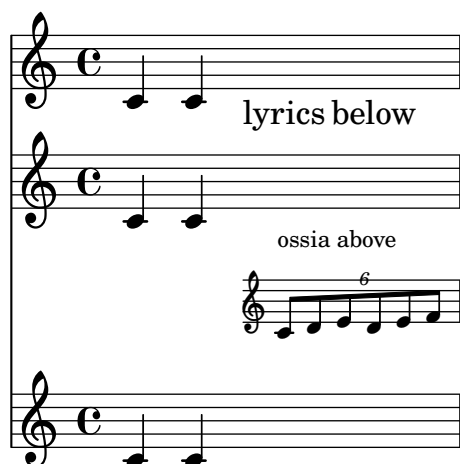


Vertically aligning ossia and lyrics

This snippet demonstrates the use of the context properties `alignBelowContext` and `alignAboveContext` to control the positioning of lyrics and ossia.

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

\relative c' <<
  \new Staff = "1" { c4 c s2 }
  \new Staff = "2" { c4 c s2 }
  \new Staff = "3" { c4 c s2 }
  { \skip 2
    <<
      \lyrics {
        \set alignBelowContext = #"1"
        lyrics4 below
      }
      \new Staff \with {
        alignAboveContext = #"3"
        fontSize = #-2
        \override StaffSymbol.staff-space = #(magstep -2)
        \remove "Time_signature_engraver"
      } {
        \tuplet 6/4 {
          \override TextScript.padding = #3
          c8[^"ossia above" d e d e f]
        }
      }
    }
  }
  >>
}
>>
```

Vertically centering paired figured bass extenders

Where figured bass extender lines are being used by setting `useBassFigureExtenders` to true, pairs of congruent figured bass extender lines are vertically centered if `figuredBassCenterContinuations` is set to true.

```
<<
\relative c' {
  c8 c b b a a c16 c b b
  c8 c b b a a c16 c b b
  c8 c b b a a c 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+>8
}
>>
```



Paper and layout

Section “Spacing issues” in *Notation Reference*

Aligning and centering instrument names

The horizontal alignment of instrument names is tweaked by changing the `Staff.InstrumentName #'self-alignment-X` property. 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 }

\score {
  \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 {
    ragged-right = ##t
```

```

    indent = 4\cm
    short-indent = 2\cm
  }
}

```

Left aligned
instrument name

Centered
instrument name

Right aligned
instrument name



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 this can be done with `\override VerticalAxisGroup.nonstaff-nonstaff-spacing.minimum-distance = ##f`.

```

\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' and 'that the' with vocal lines for ALICE and EVE. The second staff contains 'sec - ond' and 'words are' with vocal lines for ALICE and EVE. The third staff contains '...and then I' and 'Shut up, you like--' with vocal lines for EVE and ALICE. The fourth staff contains 'fool!' with a vocal line for EVE. The lyrics are written below the staves, and the vocal lines are written above the staves.

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 last page, and make the difference with the book 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.
        \on-the-fly #first-page \fromproperty #'header:copyright
      }
      \fill-line {
        %% Part tagline header field only on each part last page.
        \on-the-fly #part-last-page \fromproperty #'header:parttagline
      }
      \fill-line {
        %% Tagline header field only on book last page.
        \on-the-fly #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

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 xx}`, 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.

```
#(ly:set-option 'clip-systems)
#(define output-suffix "1")

origScore = \score {
  \relative c' {
    \new Staff \with { instrumentName = "Instrument" }
    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.

      % (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)
#(define output-suffix #f)

\book {
  \score { \origScore }
  \markup { \bold \fontsize #6 clips }
  \score {
    \lyrics {
      \markup { 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)) }
    }
  }
}

```



clips

from-2.0.1-to-4.0.1-clip.eps

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.

```

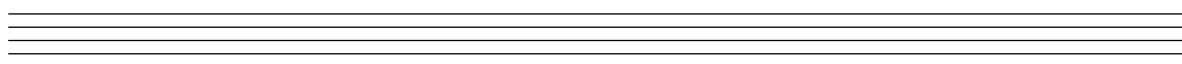
#(set-global-staff-size 20)

\score {
  {
    \repeat unfold 12 { s1 \break }
  }
  \layout {
    indent = 0\in
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Clef_engraver"
      \remove "Bar_engraver"
    }
    \context {
      \Score
      \remove "Bar_number_engraver"
    }
  }
}

% 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
}
%}

% uncomment these lines for "A4" size
%{
\paper {
  #(set-paper-size "a4")
  ragged-last-bottom = ##f
  line-width = 180
  left-margin = 15
  bottom-margin = 10
  top-margin = 10
}
%}

```



[illegible]

Demonstrating all headers

All header fields with special meanings.

```
\header {
  copyright = "copyright"
  title = "title"
  subtitle = "subtitle"
  composer = "composer"
  arranger = "arranger"
  instrument = "instrument"
  metre = "metre"
  opus = "opus"
  piece = "piece"
  poet = "poet"
  texidoc = "All header fields with special meanings."
  copyright = "public domain"
  enteredby = "jcn"
  source = "urtext"
}
```

```

\layout {
  ragged-right = ##f
}

\score {
  \relative c'' { c1 | c | c | c }
}

\score {
  \relative c'' { c1 | c | c | c }
  \header {
    title = "localtitle"
    subtitle = "localsubtitle"
    composer = "localcomposer"
    arranger = "localarranger"
    instrument = "localinstrument"
    metre = "localmetre"
    opus = "localopus"
    piece = "localpiece"
    poet = "localpoet"
    copyright = "localcopyright"
  }
}

```

	title	
	subtitle	
poet	instrument	composer
		arranger
piece		opus



localpiece	localopus
------------	-----------



Setting system separators

System separators can be inserted between systems. Any markup can be used, but `\slashSeparator` has been provided as a sensible default.

```

\paper {
  system-separator-markup = \slashSeparator
  line-width = 120
}

```

```
notes = \relative c' {  
  c1 | c \break  
  c1 | c \break  
  c1 | c  
}  
  
\book {  
  \score {  
    \new GrandStaff <<  
      \new Staff \notes  
      \new Staff \notes  
    >>  
  }  
}
```

Three systems of musical notation, each consisting of a grand staff (treble and bass clef) and a common time signature 'C'. Each system contains two measures of music. The first system has a double bar line. The second system has a double bar line and a '3' below the first measure. The third system has a double bar line and a '5' below the first measure. Each measure contains a single note on the treble staff and a single note on the bass staff.

Table of contents

A table of contents is included using `\markuplist \table-of-contents`. The TOC items are added with the `\tocItem` command.

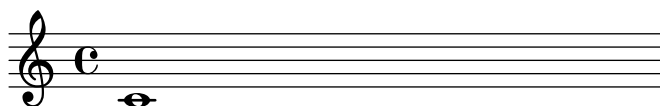
```
#(set-default-paper-size "a6")

\book {
  \markuplist \table-of-contents
  \pageBreak
  \tocItem \markup { The first score }
  \score {
    {
      c'1 \pageBreak
      \mark "A" \tocItem \markup { Mark A }
      d'1
    }
  }
  \pageBreak
  \tocItem \markup { The second score }
  \score {
    { e'1 }
    \header { piece = "Second score" }
  }
}
```

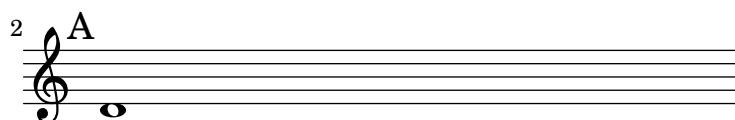
Table of Contents

The first score	2
Mark A	3
The second score	4

2



3



4

Second score



Music engraving by LilyPond 2.22.0—www.lilypond.org

Vertical aligned StaffGroups without connecting SystemStartBar

This snippet shows how to achieve vertically aligned StaffGroups with a SystemStartBar for each StaffGroup, but without connecting them.

% by Thomas Morley

```
 #(set-global-staff-size 18)
```

```
\paper {
  indent = 0
  ragged-right = ##f
  print-all-headers = ##t
}
```

```
\layout {
  \context {
    \Staff
    \consists "Mark_engraver"
    \override RehearsalMark.self-alignment-X = #LEFT
  }
  \context {
```

```

\StaffGroup
systemStartDelimiterHierarchy =
  #'(SystemStartBrace (SystemStartBracket a b))
}
\context {
  \Score
  \override SystemStartBrace.style = #'bar-line
  \omit SystemStartBar
  \override SystemStartBrace.padding = #-0.1
  \override SystemStartBrace.thickness = #1.6
  \remove "Mark_engraver"
  \override StaffGroup.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' {
  \mark \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' {
  \mark \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' {
    \mark \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.

A musical score in C major, common time (C), consisting of two staves. The lyrics are: "Wer nur den lieben Gott läßt wal-". The melody is simple, with the treble staff containing the main melody and the bass staff providing a simple harmonic accompaniment. The notes are: Treble: C4, D4, E4, F4, G4, A4, B4, C5; Bass: C3, F2, C3, F2, C3, F2, C3, F2.

Pupil: Here's my version!

A musical score in C major, common time (C), consisting of two staves. The lyrics are: "Wer nur den lieben Gott läßt wal-". The melody is more complex than the Teacher's version, with the treble staff containing the main melody and the bass staff providing a more active harmonic accompaniment. The notes are: Treble: C4, D4, E4, F4, G4, A4, B4, C5; Bass: C3, F2, C3, F2, C3, F2, C3, F2.

Teacher: Well, you simply copied and transposed a version of J.S.Bach.
Do you know this one?

A musical score in C major, common time (C), consisting of two staves. The lyrics are: "Wer nur den lieben Gott läßt wal-". The melody is more complex than the Pupil's version, with the treble staff containing the main melody and the bass staff providing a more active harmonic accompaniment. The notes are: Treble: C4, D4, E4, F4, G4, A4, B4, C5; Bass: C3, F2, C3, F2, C3, F2, C3, F2.

3

ten und hof - fet auf ihn al - le Zeit

ten und hof - fet auf ihn al - le Zeit

ten und hof - fet auf ihn al - le Zeit

Titles

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.

```
% 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!
14-01-2021



14-01-2021

Aligning and centering instrument names

The horizontal alignment of instrument names is tweaked by changing the `Staff.InstrumentName #'self-alignment-X` property. 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 }

\score {
  \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 {
  ragged-right = ##t
  indent = 4\cm
  short-indent = 2\cm
}
}

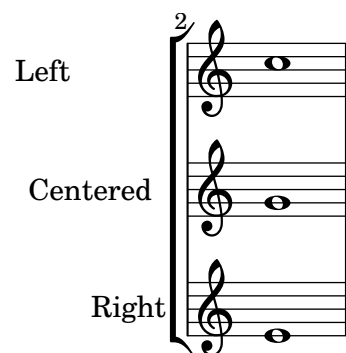
```

Left aligned
instrument name

Centered
instrument name

Right aligned
instrument name





Demonstrating all headers

All header fields with special meanings.

```
\header {
  copyright = "copyright"
  title = "title"
  subtitle = "subtitle"
  composer = "composer"
  arranger = "arranger"
  instrument = "instrument"
  metre = "metre"
  opus = "opus"
  piece = "piece"
  poet = "poet"
  texidoc = "All header fields with special meanings."
  copyright = "public domain"
  enteredby = "jcn"
  source = "urtext"
}

\layout {
  ragged-right = ##f
}

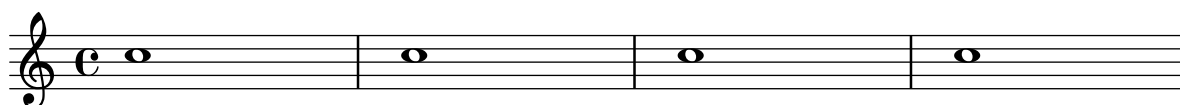
\score {
  \relative c'' { c1 | c | c | c }
}

\score {
  \relative c'' { c1 | c | c | c }
  \header {
    title = "localtitle"
    subtitle = "localsubtitle"
    composer = "localcomposer"
    arranger = "localarranger"
    instrument = "localinstrument"
    metre = "localmetre"
    opus = "localopus"
    piece = "localpiece"
    poet = "localpoet"
    copyright = "localcopyright"
  }
}
```

	title	
	subtitle	
poet	instrument	composer
		arranger
piece		opus



localpiece	localopus
------------	-----------



Outputting the version number

By putting the output of `lilypond-version` into a lyric, it is possible to print the version number of LilyPond in a score, or in a document generated with `lilypond-book`. Another possibility is to append the version number to the doc-string, in this manner:

```
\score {
  \new Lyrics {
    \override Score.RehearsalMark.self-alignment-X = #LEFT
    \mark #(string-append "Processed with LilyPond version " (lilypond-version))
    s2
  }
}
```

Processed with LilyPond version 2.22.0

Spacing

Section “Spacing issues” in *Notation Reference*

Adjusting lyrics vertical spacing

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
}

```



Page label

Page labels may be placed inside music or at top-level, and referred to in markups.

```

#(set-default-paper-size "a6")

#(define-markup-command (toc-line layout props label text)
  (symbol? markup?)
  (interpret-markup layout props
    (markup #:fill-line (text #:page-ref label "8" "?"))))

\book {
  \markup \huge \fill-line { \null Title Page \null }

  \pageBreak

  \label #'toc
  \markup \column {
    \large \fill-line { \null Table of contents \null }
    \toc-line #'toc "Table of contents"
    \toc-line #'firstScore "First Score"
    \toc-line #'markA "Mark A"
    \toc-line #'markB "Mark B"
    \toc-line #'markC "Mark C"
    \toc-line #'unknown "Unknown label"
  }

  \pageBreak

  \label #'firstScore
  \score {
    \new Staff \relative c' {
      c2 c
      \mark \markup {
        A (page \concat { \page-ref #'markA "0" "?" } ) }
      } \label #'markA
      c2 c
      \pageBreak
      \mark "B" \label #'markB
      d2 d
    }
  }
}

```

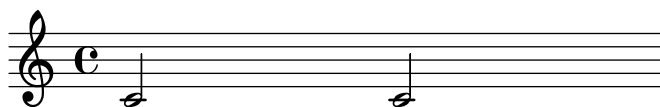


```

d2 d
\once \override Score.RehearsalMark.break-visibility =
  #begin-of-line-invisible
\mark "C" \label #'markC
}
\header { piece = "First score" }
}
}

```

	Title	Page
2	Table of contents	
	Table of contents	2
	First Score	3
	Mark A	3
	Mark B	4
	Mark C	4
	Unknown label	?
		3
	First score	



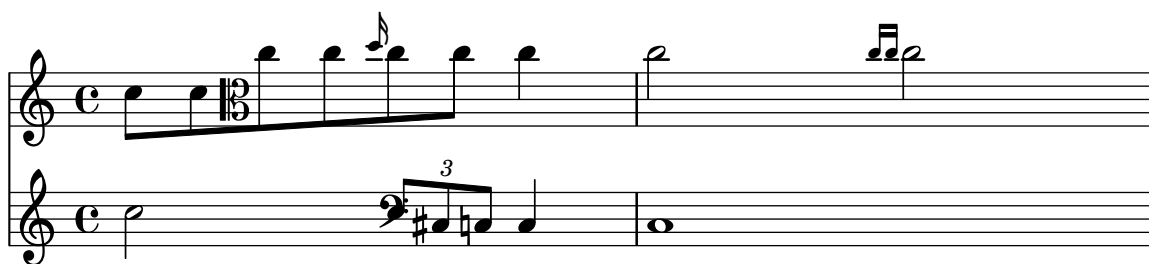


Music engraving by LilyPond 2.22.0—www.lilypond.org

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 = #(ly:make-moment 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
  }
>>
```



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
}
```



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.

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

\relative c' <<
  \new Staff = "1" { c4 c s2 }
  \new Staff = "2" { c4 c s2 }
  \new Staff = "3" { c4 c s2 }
  { \skip 2
```

```

<<
  \lyrics {
    \set alignBelowContext = #"1"
    lyrics4 below
  }
  \new Staff \with {
    alignAboveContext = #"3"
    fontSize = #-2
    \override StaffSymbol.staff-space = #(magstep -2)
    \remove "Time_signature_engraver"
  } {
    \tuplet 6/4 {
      \override TextScript.padding = #3
      c8[~"ossia above" d e d e f]
    }
  }
>>
}
>>

```

The image displays a musical score with three staves. The top staff contains two quarter notes, with the text "lyrics below" positioned directly beneath them. The middle staff also contains two quarter notes, followed by a triplet of eighth notes. Above this triplet is the text "ossia above". The bottom staff contains two quarter notes. The notation is in common time (C) and the key signature has one flat (B-flat).

MIDI

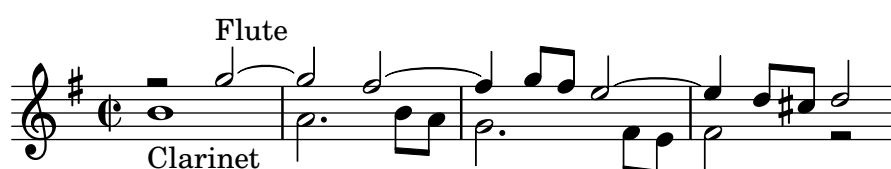
Section “Creating MIDI output” in *Notation Reference*

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 it a specific value so that it can be used to affect MIDI output.

The dynamic mark `\rfz` is assigned a value of 0.9.

```
#(define (myDynamics dynamic)
  (if (equal? dynamic "rfz")
    0.9
    (default-dynamic-absolute-volume dynamic)))

\score {
  \new Staff {
    \set Staff.midiInstrument = #"cello"
    \set Score.dynamicAbsoluteVolumeFunction = #myDynamics
    \new Voice {
      \relative {
        a'4\pp b c-\rfz
      }
    }
  }
  \layout {}
  \midi {}
}
```



Customized drum notation in printed and MIDI output

Customized drum “pitch” names (suitable for a custom drum style, for example) may be used both in printed and MIDI output by defining such variables as `drumPitchNames`, `drumStyleTable` and `midiDrumPitches`, as demonstrated here. In short, this snippet:

- defines some "pitch" names
- defines how they will be rendered
- tell LilyPond to use them for layout
- assigns pitches to the names
- tell LilyPond to use them for MIDI output

```
%% This snippet tries to amend
```

%% NR 2.5.1 Common notation for percussion - Custom percussion staves

%% <http://lilypond.org/doc/v2.18/Documentation/notation/common-notation-for-percussion#custo>

[illegible]

%%

```
%% To use custom drum-pitch-names for your score and midi you need to follow
```

```
%% this route:
```

%%

0/0/0/0/0/0/0/0/0/0/0/0/0/0/0

```
%% LAYOUT:
```

0/0/0/0/0/0/0/0/0/0/0/0/0/0/0/

%%

```
%% (1) Define a name and put it in `drumPitchNames'
```

%% This can be done at toplevel with

```
%% drumPitchNames.my-name = #'my-name
```

```
%% It's possible to add an alias as well.
```

```
%% (2) Define how it should be printed
```

```
%% Therefore put them into a toplevel-list, where each entry should look:
```

```
%% (my-name
```

```
%% note-head-style-or-default
```

```
%% articulation-string-or-#f
```

```
%%                                staff-position)
```

%% Example:

```
%%          # (define my-style
```

 $\frac{0}{0} \frac{0}{0}$

```
%% (my-name default "tenuto" -1)
```

%% ; . . .

%%))

```
%% (3) Tell LilyPond to use this custom-definitions, with
```

```
%%      drumStyleTable = #(alist->hash-table my-style)
```

```
%% in a \layout or \with
```

%%

```
%%      Now we're done for layout, here a short, but complete example:
```

```
%% \new DrumStaff
```

```
%% \with { drumStyleTable = #(alist->hash-table my-style) }
```

```
%% \drummode { my-name }
```

%%

0/0/0/0/0/0/0/0/0/0/0/0/0/0/0

```
%% MIDI:
```

%%%%%%%%%

```

%%
%% (1) Again at toplvel, assign a pitch to your custom-note-name
%%      midiDrumPitches.my-name = ges
%%      Note that you have to use the name, which is in drumPitchNames, no alias
%% (2) Tell LilyPond to use this pitch(es), with
%%      drumPitchTable = #(alist->hash-table midiDrumPitches)
%%
%%      Example:
%%      \score {
%%        \new DrumStaff
%%        \with {
%%          drumStyleTable = #(alist->hash-table my-style)
%%          drumPitchTable = #(alist->hash-table midiDrumPitches)
%%        }
%%        \drummode { my-name4 }
%%      \layout {}
%%      \midi {}
%%    }
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% TESTING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% To test whether all is fine, run the following sequence in terminal:
%%      lilypond my-file.ly
%%      midi2ly my-file.midi
%%      gedit my-file-midi.ly
%%
%% Which will do:
%% 1. create pdf and midi
%% 2. transform the midi back to a .ly-file
%%    (note: midi2ly is not always good in correctly identifying enharmonic pitches)
%% 3. open this file in gedit (or use another editor)
%% Now watch what you've got.
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% FULL EXAMPLE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

drumPitchNames.dbass      = #'dbass
drumPitchNames.dba       = #'dbass  % 'db is in use already
drumPitchNames.dbassmute = #'dbassmute
drumPitchNames.dbm       = #'dbassmute
drumPitchNames.do        = #'dopen
drumPitchNames.dopenmute = #'dopenmute
drumPitchNames.dom       = #'dopenmute
drumPitchNames.dslap     = #'dslap
drumPitchNames.ds        = #'dslap
drumPitchNames.dslapmute = #'dslapmute
drumPitchNames.dsm       = #'dslapmute

```



```

#(define djembe
  '((dbass      default  #f          -2)
    (dbassmute  default  "stopped"  -2)
    (dopen      default  #f          0)
    (dopenmute  default  "stopped"  0)
    (dslap      default  #f          2)
    (dslapmute  default  "stopped"  2)))

midiDrumPitches.dbass = g
midiDrumPitches.dbassmute = fis
midiDrumPitches.dopen = a
midiDrumPitches.dopenmute = gis
midiDrumPitches.dslap = b
midiDrumPitches.dslapmute = ais

one = \drummode { r4 dba4 do ds r dbm dom dsm }

\score {
  \new DrumStaff
  \with {
    \override StaffSymbol.line-count = #3
    instrumentName = #"Djembe "
    drumStyleTable = #(alist->hash-table djembe)
    drumPitchTable = #(alist->hash-table midiDrumPitches)
  }
  \one
  \layout {}
  \midi {}
}

```



Demo MidiInstruments

Problem: How to know which `midiInstrument` would be best for your composition?

Solution: A LilyPond demo file.

```
\header {
  title = "Demo of all midi sounds"
  arranger = "Myself "
}

baseMelody = \relative c' {
  c4.\mf g c16 b' c d
  e16 d e f g4 g'4 r
  R1
}

melody = {
  \tempo 4 = 150
  \baseMelody
```

```
}
```

```
\score {
  \new Staff <<
    \new Voice \melody
  >>
  \layout { }
}
```

```
\score {
  \new Staff <<
    \new Voice {
      r\mf
      \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
      \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
      \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
      \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 = #"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
    }
  >>
}
```

```
\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
\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
\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
\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
\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
\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
\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
\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
\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
\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
\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 { }
}

```

Demo of all midi sounds

Myself



Replacing default MIDI instrument equalization

The default MIDI instrument equalizer can be replaced by setting the `instrumentEqualizer` property in the `Score` context to a user-defined Scheme procedure that uses a MIDI instrument name as its argument along with a pair of fractions indicating the minimum and maximum volumes respectively to be applied to that specific instrument.

The following example sets the minimum and maximum volumes for flute and clarinet respectively.

```
#(define my-instrument-equalizer-alist '())

#(set! my-instrument-equalizer-alist
  (append
    '(
      ("flute" . (0.7 . 0.9))
      ("clarinet" . (0.3 . 0.6)))
    my-instrument-equalizer-alist))

#(define (my-instrument-equalizer s)
  (let ((entry (assoc s my-instrument-equalizer-alist)))
    (if entry
      (cdr entry))))

\score {
  <<
    \new Staff {
      \key g \major
      \time 2/2
      \set Score.instrumentEqualizer = #my-instrument-equalizer
      \set Staff.midiInstrument = "flute"
      \new Voice \relative {
        r2 g'\mp g fis~
        4 g8 fis e2~
        4 d8 cis d2
      }
    }
    \new Staff {
      \key g \major
      \set Staff.midiInstrument = "clarinet"
      \new Voice \relative {
        b'1\p a2. b8 a
        g2. fis8 e
        fis2 r
      }
    }
  >>
  \layout { }
  \midi { }
}
```

Two staves of music in G major (one sharp) and 4/4 time. The top staff begins with a mezzo-piano (*mp*) dynamic. The bottom staff begins with a piano (*p*) dynamic. The music consists of a four-measure phrase.

Staff 1 (Top): Treble clef, G major. Measures: 1. Quarter rest, quarter G4. 2. Quarter A4, quarter B4. 3. Quarter C5, quarter B4. 4. Quarter A4, quarter G4.

Staff 2 (Bottom): Treble clef, G major. Measures: 1. Half rest. 2. Quarter G4, eighth A4, eighth B4. 3. Quarter C5, eighth B4, eighth A4. 4. Quarter G4, half rest.

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.

```
\include "gregorian.ly"

chant = \relative c' {
  \set Score.timing = ##f
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g) a2 \finalis
}

verba = \lyricmode {
  Lo -- rem ip -- sum do -- lor sit a -- met
}

\score {
  \new Staff <<
    \new Voice = "melody" \chant
    \new Lyrics = "one" \lyricsto melody \verba
  >>
  \layout {
    \context {
      \Staff
      \remove "Time_signature_engraver"
      \remove "Bar_engraver"
      \hide Stem
    }
    \context {
      \Voice
      \override Stem.length = #0
    }
    \context {
      \Score
      barAlways = ##t
    }
  }
}
```



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 = #(ly:make-moment 1/2)
    }
    \context {
        \Staff
        \remove "Time_signature_engraver"
    }
}
} % End score
} % End markup

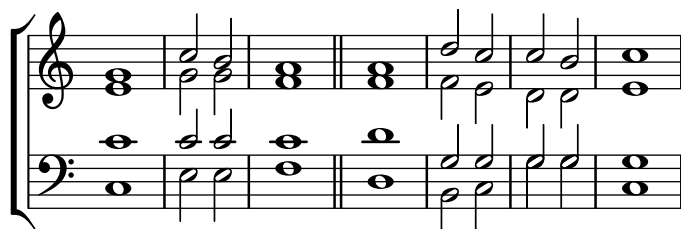
\markup {
    \fill-line {
        \column {
            \left-align {
                \null \null \null
            }
            \line {
                \fontsize #5 0
                \fontsize #3 come
                let us \bold sing | unto \dot the | Lord : let
            }
            \line {
                us heartily
                \concat { re \bold joice }
                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 when each line starts and ends with a partial measure. It also shows how to add the verses as stand-alone text under the music.

```

Timeline = {
  \time 4/4
  \tempo 4=96
  \partial 2
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||" \break
  s2 | s1 | s2 \breathe s2 | s1 | s2 \bar "||"
}

SopranoMusic = \relative g' {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

```

```

}

AltoMusic = \relative c' {
  d4 d | d d d d | d d d d | d d d d | d2
  d4 d | d d d d | d d d d | d d d d | d2
}

TenorMusic = \relative a {
  b4 b | b b b b | b b b b | b b b b | b2
  b4 b | b b b b | b b b b | b b b b | b2
}

BassMusic = \relative g {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

global = {
  \key g \major
}

\score { % Start score
  <<
    \new PianoStaff << % Start pianostaff
      \new Staff << % Start Staff = RH
        \global
        \clef "treble"
        \new Voice = "Soprano" << % Start Voice = "Soprano"
          \Timeline
          \voiceOne
          \SopranoMusic
        >> % End Voice = "Soprano"
        \new Voice = "Alto" << % Start Voice = "Alto"
          \Timeline
          \voiceTwo
          \AltoMusic
        >> % End Voice = "Alto"
      >> % End Staff = RH
    \new Staff << % Start Staff = LH
      \global
      \clef "bass"
      \new Voice = "Tenor" << % Start Voice = "Tenor"
        \Timeline
        \voiceOne
        \TenorMusic
      >> % End Voice = "Tenor"
      \new Voice = "Bass" << % Start Voice = "Bass"
        \Timeline
        \voiceTwo
        \BassMusic
      >> % End Voice = "Bass"
    >> % End Staff = LH
  }
}

```

```

    >> % End pianostaff
  >>
} % End score

\markup {
  \fill-line {
    ""
    {
      \column {
        \left-align {
          "This is line one of the first verse"
          "This is line two of the same"
          "And here's line three of the first verse"
          "And the last line of the same"
        }
      }
    }
    ""
  }
}

\paper { % Start paper block
  indent = 0 % don't indent first system
  line-width = 130 % shorten line length to suit music
} % End paper block

```



This is line one of the first verse
 This is line two of the same
 And here's line three of the first verse
 And the last line of the same

Jazz combo template

This is quite an advanced template, for a jazz ensemble. Note that all instruments are notated in `\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 in the documentation
\paper {
  paper-width = 130
}
%#(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
  \nsl
}
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
  \nsl
  c1
}
gtrHarmony = \chordmode {
  \jazzChords
  s1 c2:min7+ d2:maj9
}
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 %%%%%%%%%%%

\score {
  <<
    \new StaffGroup = "horns" <<
      \new Staff = "trumpet" \with { instrumentName = "Trumpet" }
      \trumpet
      \new Staff = "altosax" \with { instrumentName = "Alto Sax" }
      \altoSax
      \new ChordNames = "barichords" \with { instrumentName = "Trumpet" }
      \bariHarmony
      \new Staff = "barisax" \with { instrumentName = "Bari Sax" }
      \bariSax
      \new Staff = "trombone" \with { instrumentName = "Trombone" }
      \trombone
    >>

    \new StaffGroup = "rhythm" <<
      \new ChordNames = "chords" \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^{Δ} $C\#m^7$ Solo

Cm^{Δ} $D^{\Delta 9}$

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 }

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

altoILyrics = \sopranoLyrics

altoIIILyrics = \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 }

\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 { \pianoRHMus }
  \new Staff { \pianoLHMus }
>>

\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
>>
>>
\layout { }
}
```

A musical score template for a symphony orchestra and vocal ensemble. The score is written for 15 parts: 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 key signature is one sharp (F#) and the time signature is 4/4. The score is divided into two systems. The first system contains the Flute, Clarinet in B \flat , Horn in F, Trumpet in C, Percussion, Piano, and the vocal parts (Soprano, Alto I, Alto II, Tenor). The second system contains the Violin I, Violin II, Viola, Cello, and Double Bass. The vocal parts have lyrics: "Lyr - ics" for Soprano, Alto I, and Tenor; "Lyr - ics" for Alto II; and "Ah - ah" for the Tenor. The piano part has a simple melody in the right hand and a simple bass line in the left hand. The instrumental parts have simple harmonic support. The score is marked with a page number 8 at the bottom left.

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

SATB choir template (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
  >>
}

```



Score for diatonic accordion

A template to write a score for a diatonic accordion.

- There is a horizontal staff indicating if the accordion must be pushed (thick line) or pulled (thin line)
- There is a small rhythmic staff with lyrics that describes the bass buttons to press. The bar lines are made from gridlines
- The tabulator staff for diatonic accordions shows the geographic position of the buttons and not (as for every other instrument) the pitch of the notes; the keys on the melody-side of the accordion are placed in three columns and about 12 rows

In the tabulator staff notation the outermost column is described with notes between lines, the innermost column is described with notes between lines and a cross as accidental, and the middle column is described with notes on a line, whereby the row in the middle is represented on the middle line in the staff.

Some words to transpose piano notes to the diatonic accordion:

1. Every diatonic accordion is built for some keys only (for example, for the keys of C major and F major), so it is important to transpose a piano melody to match one of these keys.

Transpose the source code, not only the output because this code is required later on to translate it once more to the tabulator staff. This can be done with the command `displayLilyMusic`.

2. You have to alternate the push- and pull-direction of the accordion regularly. If the player has a too long part to pull the accordion gets broken. On the other hand, some harmonies are only available in one direction. Considering this, decide which parts of the melody are the push-parts and which the pull-parts.

3. For each pull- or push-part translate the piano notes to the according tabulature representation.

```
verse = \lyricmode { Wie gross bist du! Wie gross bist du! }

harmonies = \new ChordNames \chordmode {
  \germanChords
  \set chordChanges = ##t
  bes8 bes8 bes8
  es2 f
  bes1
}

NoStem = { \hide Stem }
NoNoteHead = \hide NoteHead
ZeroBeam = \override Beam.positions = #'(0 . 0)

staffTabLine = \new Staff \with {
  \remove "Time_signature_engraver"
  \remove "Clef_engraver"
} {
  \override Staff.StaffSymbol.line-positions = #'(0)
  % Shows one horizontal line. The vertical line
  % (simulating a bar-line) is simulated with a gridline
  \set Staff.midiInstrument = #"choir aahs"
  \key c \major
  \relative c''
  {
    % disable the following line to see the noteheads while writing the song
    \NoNoteHead
    \override NoteHead.no-ledgers = ##t

    % The beam between 8th-notes is used to draw the push-line
    %How to fast write the push-lines:
    % 1. write repeatedly 'c c c c c c c c |' for the whole length of the song
    % 2. uncomment the line \NoNoteHead
    % 3. compile
    % 4. Mark the positions on which push/pull changes.
    %     In the score-picture click on the position
    %         the push- or pull-part starts
    %         (on the noteHead, the cursor will change to a hand-icon).
    %     The cursor in the source code will jump just at this position.
    % a) If a push-part starts there, replace the 'c' by an 'e['
    % b) If a pull-part starts there, replace the 'c' by an 's'
    % 5. Switch into 'overwrite-mode' by pressing the 'ins' key.
    % 6. For the pull-parts overwrite the 'c' with 's'
```

```

% 7. For every push-part replace the last 'c' with 'e]'
%      8. Switch into 'insert-mode' again
% 9. At last it should look like e.g.
%      (s s e[ c | c c c c c c c c | c c c c c c e] s s)
% 10. re-enable the line \NoNoteHead
\autoBeamOff
\ZeroBeam
s8 s s e[ c c c c c c e] | s s s s s
}
}

% Accordion melody in tabulator score
% 1. Place a copy of the piano melody below
% 2. Separate piano melody into pull- and push-parts
%      according to the staffTabLine you've already made
% 3. For each line: Double the line. Remark the 1st one
%      (Keeps unchanged as reference) and then change the second
%      line using the transformation paper
%      or the macros 'conv2diaton push.bsh' and 'conv2diaton pull.bsh'
% Tips:
% - In jEdit Search & Replace mark the Option 'Keep Dialog'

AccordionTabTwoCBesDur = {
  % pull 1
  %<f' bes'>8 <f' a'>8 <d' bes'>8 |
  <g'' a''>8 <g'' b''>8 <e'' a''>8 |
  % push 2
  %<g' c''>4 <f' d''> <g' ees''> <f' a'> |
  <g'' a''>4 <d'' eisis''> <g'' bisis''> <d'' f''> |
  % pull 3
  % <f' bes'>2 r8 }
  <g'' a''>2 r8 }

AccordionTab= { \dynamicUp
  % 1. Place a copy of the piano melody above
  % 2. Separate piano melody into pull- and push-parts
  %      according to the staffTabLine you've already made
  % 3. For each line: Double the line. Remark the 1st one
  %      (Keeps unchanged as reference) and then
  %      change the second line using the transformation paper
  % Tips:
  % - In jEdit Search & Replace mark the Option 'Keep Dialog'
  % -
  \AccordionTabTwoCBesDur
}

\layout {
  \context {
    \Score
    % The vertical line (simulating a bar-line) in

```

```

    % the staffBassRhythm is a gridline
    \consists "Grid_line_span_engraver"
  }
  \context {
    \Staff
    \consists "Grid_point_engraver"
    gridInterval = #(ly:make-moment 4/4) % 4/4 - tact. How many beats per bar
    % The following line has to be adjusted O-F-T-E-N.
    \override GridPoint.Y-extent = #'(-2 . -21)
  }
  \context {
    \ChoirStaff
    \remove "System_start_delimiter_engraver"
  }
}

staffVoice = \new Staff = astaffvoice {
  \time 4/4
  \set Staff.instrumentName = "Voice"
  \set Staff.midiInstrument = "voice oohs"
  \key bes \major
  \partial 8*3
  \clef treble
  {
    \context Voice = "melodyVoi"
    {
      <f' bes'>8 <f' a'>8 <d' bes'>8 |
      <g' c''>4 <f' d''> <g' es''> <f' a'> |
      <f' bes'>2 r8
    }
    \bar "|."
  }
}

staffAccordionMel =
\new Staff \with { \remove "Clef_engraver" } {
  \accidentalStyle forget %Set the accidentals (Vorzeichen) for each note,
  %do not remember them for the rest of the measure.
  \time 4/4
  \set Staff.instrumentName="Accordion"
  \set Staff.midiInstrument="voice oohs"
  \key c \major
  \clef treble
  { \AccordionTab \bar "|." }
}

AltOn =
#(define-music-function (mag) (number?)
  #{ \override Stem.length = #(* 7.0 mag)
    \override NoteHead.font-size =
  #(inexact->exact (* (/ 6.0 (log 2.0)) (log mag))) #})

```

```

AltOff = {
  \revert Stem.length
  \revert NoteHead.font-size
}

BassRhytm = {s4 s8 | c2 c2 | c2 s8 }
LyricBassRhythmI= \lyricmode { c b | c }

staffBassRhytm =
\new Staff = staffbass \with { \remove "Clef_engraver" } {
  % This is not a RhythmicStaff because it must be possible to append lyrics.

  \override Score.GridLine.extra-offset = #'( 13.0 . 0.0 ) % x.y
  \override Staff.StaffSymbol.line-positions = #'( 0 )
  % Shows one horizontal line. The vertical line
  % (simulating a bar-line) is simulated by a grid
  % Search for 'grid' in this page to find all related functions
  \time 4/4
  {
    \context Voice = "VoiceBassRhytm"
    \stemDown \AltOn #0.6
    \relative c''
    {
      \BassRhytm
    }
    \AltOff
    \bar "|."
  }
}

\score {
  \new ChoirStaff <<
    \harmonies
    \staffVoice
    \context Lyrics = "lmelodyVoi"
    \with { alignBelowContext = astaffvoice }
    \lyricsto melodyVoi \verse
    \staffAccordionMel
    \staffTabLine
    \staffBassRhytm
    \context Lyrics = "lBassRhytmAboveI"
    \with { alignAboveContext = staffbass }
    \lyricsto VoiceBassRhytm \LyricBassRhythmI
  >>
}

```

Voice

Wie gross bist du! Wie gross bist du!

Accordion

B E \flat F B

c b c

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 { }
}
```

A C

Aaa Bee Cee Dee

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

```

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 { }
}

```

Violin 1

Violin 2

Viola

Cello

String quartet template with separate parts

The “String quartet template” snippet produces a nice string quartet, but what if you needed to print parts? This new template demonstrates how to use the `\tag` feature to easily split a piece into individual parts.

You need to split this template into separate files; the filenames 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 appropriate part.

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 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 "|."
  }
}
```

```

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)

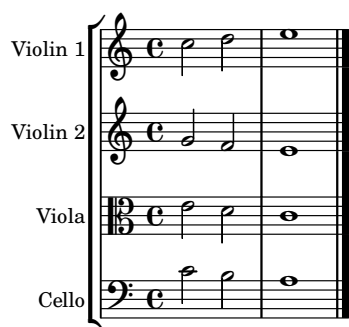
\include "piece.ly"
\score {
  \keepWithTag #'vla \music
  \layout { }
}

% vlc.ly
% (This is the Cello part file)

\include "piece.ly"
\score {
  \keepWithTag #'vlc \music
  \layout { }
}

%}

```



Vocal ensemble template with automatic piano reduction

This template adds an automatic piano reduction to the standard SATB vocal score demonstrated in “Vocal ensemble template”. This 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 will 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
      }
    >>

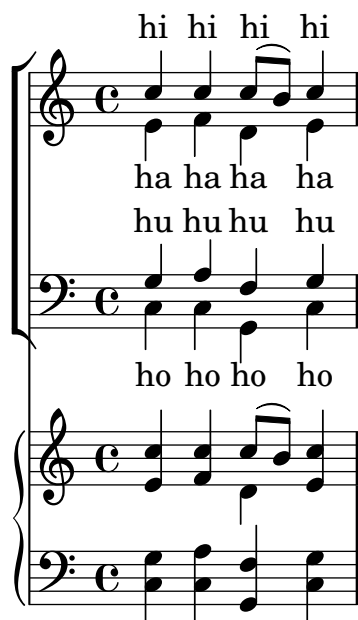
    \new Staff = "men" <<
      \clef bass

```

```

\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 >>
  >>
>>
>>
>>
}

```



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 which 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
}
}
}

```



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
  >>
}

```

