New features in 2.20 since 2.18

New for musical notation

Displaying pitch improvements

- Pitches that have a sharp or flat in their name now need to be hyphenated;

  ```
  \key a-flat \major
  ```

  instead of:

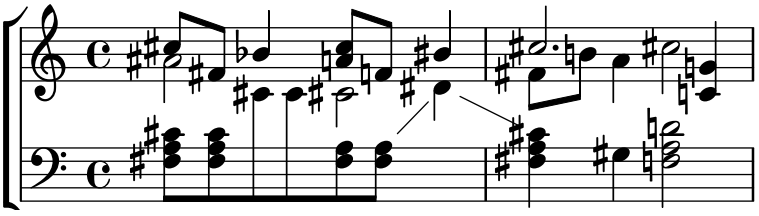
  ```
  \key aflat \major
  ```

  Pitches that contain *double* sharps or flats in their name, however, do not need a second hyphen. For example using the Dutch notation `cisis`:

  ```
  \key c-sharpsharp \major
  ```

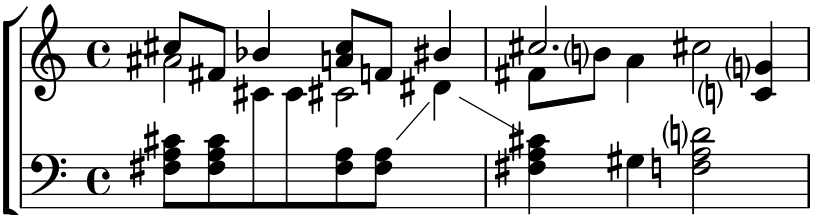
- Accidental rules can now be defined *across* `ChoirStaff` contexts.
- Two new accidental rules have been added. Both combine the characteristics of `modern-voice`, `piano` and their equivalents:

  `choral`



  This is the now the default accidental style for `ChoirStaff`.

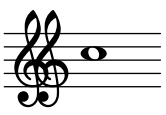
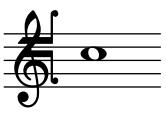
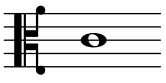
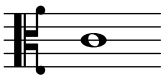
  `choral-cautionary`



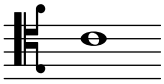
  The same as `choral` but with the extra accidentals typeset as cautionaries instead.

  Also see Section "Automatic accidentals" in *Notation Reference*.

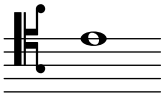
- Four new clef glyphs are now available; 'GG' (double-G), 'Tenor G', 'varC' plus related tessitura and 'Varpercussion':

Example	**Output**	**Example**	**Output**
`\clef GG`		`\clef tenorG`	
`\clef varC`		`\clef altovarC`	

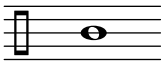
`\clef tenorvarC`



`\clef baritonevarC`



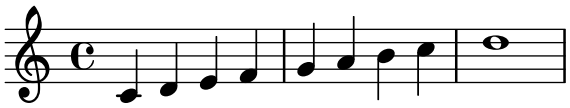
`\clef varpercussion`



Also see Section "Clef styles" in *Notation Reference*.

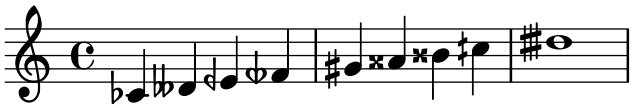
- French note names are now explicitly defined – previously they were aliased to Italian note names. The *d* pitch may be entered as either `re` or `ré`.

```
\language "français"
do ré mi fa | sol la si do | ré1
```



Double sharps are entered using an `x` suffix.

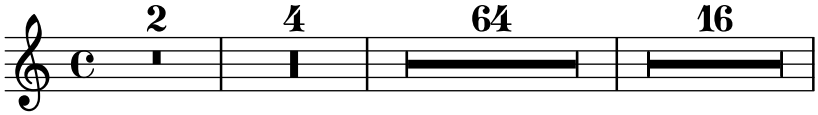
```
\language "français"
dob, rebb misb fabsb | sold ladd six dosd | rédsd1
```



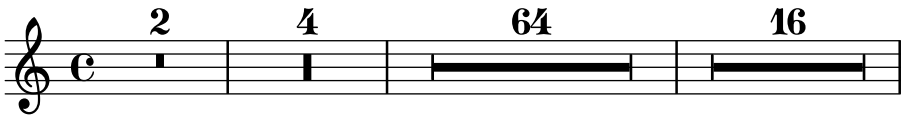
Rhythm improvements

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

```
\compressFullBarRests
R1*2 R1*4 R1*64 R1*16
```



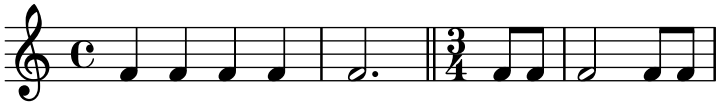

```
\compressFullBarRests
\override Staff.MultiMeasureRest.space-increment = 2.5
R1*2 R1*4 R1*64 R1*16
```



- Improvements to the `\partial` command have been made when used with parallel music and/or multiple contexts.
- It is now possible to change the time signature mid-measure by using both the `\time` and `\partial` commands together.

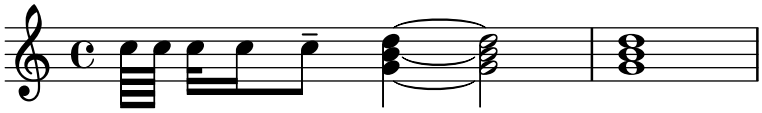
```
f f f f | f2. \bar "||"
\time 3/4 \partial 4
```

```
f8 8 | f2 f8 f |
```

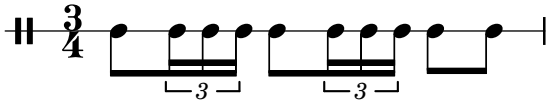


- Isolated durations in music now stand for unpitched notes. Pitches are taken from the preceding note or chord. This is especially convenient for specifying rhythms in both music and scheme functions and can help improve the readability of LilyPond source files.

```
c64[ 64] 32 16 8^- <g b d>4~ 2 | 1
```



```
\new DrumStaff \with { \override StaffSymbol.line-count = 1 }
\drummode {
  \time 3/4
  tambourine 8 \tuplet 3/2 { 16 16 16 }
             8 \tuplet 3/2 { 16 16 16 } 8 8 |
}
```

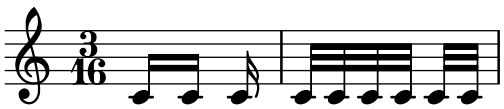


- Beaming exceptions can now be constructed using the simpler `\beamExceptions` scheme function. Previously, this would have required writing:

```
\set Timing.beamExceptions =
#'(                          ;start of alist
   (end .                    ;entry for end of beams
    (                        ;start of alist of end points
     ((1 . 32) . (2 2 2))    ;rule for 1/32 beams -- end each 1/16
    )))

\time #'(2 1) 3/16
c16 c c
\repeat unfold 6 { c32 }
```

With the new `\beamExceptions` scheme function, this becomes:

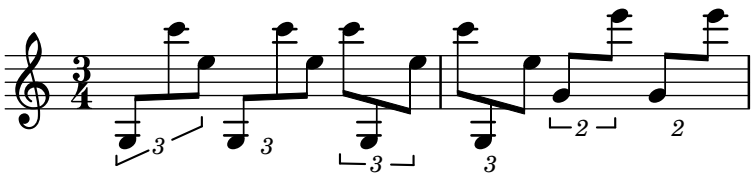
```
\set Timing.beamExceptions =
  \beamExceptions { 32[ 32] 32[ 32] 32[ 32] }

\time #'(2 1) 3/16
c16 c c |
\repeat unfold 6 { c32 } |
```



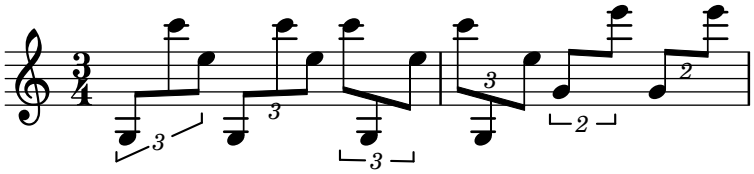
with multiple exceptions separated by bar checks. Note that writing the exception pattern without pitches is convenient but not mandatory (also see the previous documented rhythm improvement – *Isolated durations in music now stand for unpitched notes.*

- The positioning of tuplet numbers for kneed beams has been improved. Previously, tuplet numbers were placed according to the position of the tuplet bracket, even if the bracket was not printed. This could lead to tuplet numbers being 'stranded'.

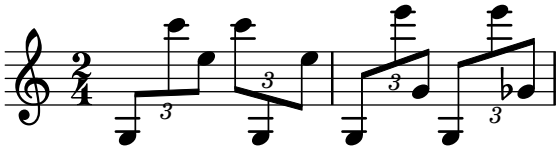
  Previously:



  Now, when the bracket is not drawn, tuplet numbers are positioned closer.

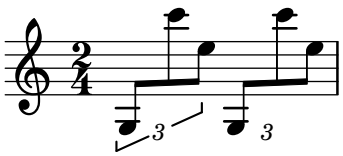


- Collision detection for the kneed beam tuplet numbers has also been added, shifting the offset horizontally if the number is too close to an adjoining note column (but still preserving the number's vertical distance). In the event of a collision – for example with an accidental – the tuplet number will be shifted vertically instead. If the tuplet number is itself too large to fit within the available space, the original, 'bracket-based', positioning system will be used instead.



  The original kneed-beam tuplet behavior is still available with a new, `knee-to-beam` property for the `TupletNumber` layout object.

```
\time 2/4
\override Beam.auto-knee-gap = 3
\override TupletNumber.knee-to-beam = ##f
\override TupletBracket.bracket-visibility = ##t
\tuplet 3/2 4 { g8 c'' e, }
\once \override TupletBracket.bracket-visibility = ##f
\tuplet 3/2 4 { g,,8 c'' e, }
```



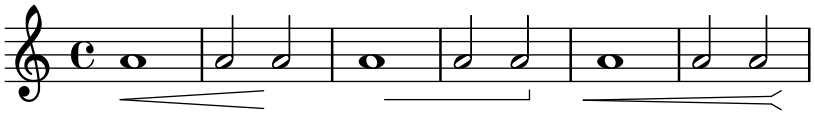
Expressive mark improvements

- The ends of hairpins may now be fine-tuned using the `shorten-pair` grob property. This previously only affected text-spanners (e.g. `TupletBracket` and `OttavaBracket`).

  Positive and negative values offset right and left respectively.

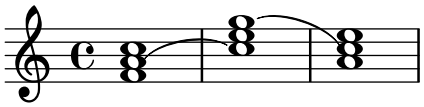
```
\once \override Hairpin.shorten-pair = #'(0 . 2)
```

```
a1\< | a2 a\!

\once \override Hairpin.shorten-pair = #'(2 . 0)
\once \override Hairpin.stencil = #constante-hairpin
a1\< | a2 a\!

\once \override Hairpin.shorten-pair = #'(-1 . -1)
\once \override Hairpin.stencil = #flared-hairpin
a1\< | a2 a\!
```

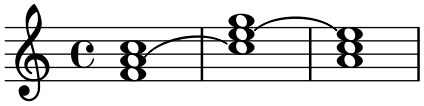


- Individual slurs and phrasing slurs may now be started from an explicit note within a chord.

```
<f a( c>1 | <c') e g(> | <a c) e>
```



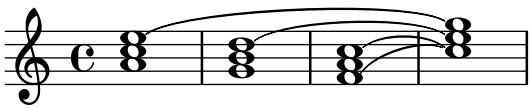
```
<f( a\( c>1 | <c'\) e\( g> | <a c e\)>
```



- A new command `\=X` has been added – where 'X' can be any non-negative integer or symbol – so that a specific 'id' can be assigned to the start and end of slurs and phrasing slurs.

  This is useful when simultaneous slurs are required or if one slur overlaps another or when nesting short slurs within a longer one.

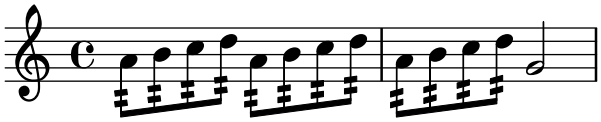
```
<a c e\=7\(>1 | <g b d\=£(> |
<f\=A( a c\="foo"(> | <c'\="foo")\=A) e\=£) g\=7\)> |
```



  Also see Section "Expressive marks as curves" in *Notation Reference*.

Repeat notation improvements

- The visual style of tremolo slashes (shape, style and slope) is now more finely controlled.

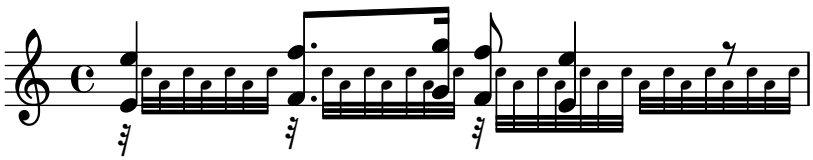


- The music function `\unfoldRepeats` can now take an optional argument-list specifying which type(s) of repeated music should be unfolded. Possible entries are `percent`, `tremolo`, `volta`. If the optional argument-list is unspecified, `repeated-music` will be used, unfolding all.

Staff notation improvements

- A new command `\magnifyStaff` has been added which scales staff sizes, staff lines, bar lines, beamlets and horizontal spacing generally at the `Staff` context level. Staff lines are prevented from being scaled smaller than the default since the thickness of stems, slurs, and the like are all based on the staff line thickness.
- A new command `\magnifyMusic` has been added, which allows the notation size to be changed without changing the staff size, while automatically scaling stems, beams, and horizontal spacing.

```
\new Staff <<
  \new Voice \relative {
    \voiceOne
    <e' e'>4 <f f'>8. <g g'>16 <f f'>8 <e e'>4 r8
  }
  \new Voice \relative {
    \voiceTwo
    \magnifyMusic 0.63 {
      \override Score.SpacingSpanner.spacing-increment = #(* 1.2 0.63)
      r32 c'' a c a c a c r c a c a c a c
      r c a c a c a c a c a c a c a c
    }
  }
>>
```



- A new command, `\RemoveAllEmptyStaves`, has been made available, which acts exactly like `\RemoveEmptyStaves`, except for also removing empty staves on the first system in a score.
- A new markup command `\justify-line` has been added. Similar to the `\fill-line` markup command except that instead of setting *words* in columns, the `\justify-line` command balances the whitespace between them ensuring that when there are three or more words in a markup, the whitespace is always consistent.

```
\markup \fill-line {oooooo oooooo oooooo oooooo}
\markup \fill-line {ooooooooo ooooooooo oo ooo}
```

oooooo oooooo oooooo oooooo

ooooooooo oooooooo oo ooo

```
\markup \justify-line {oooooo oooooo oooooo oooooo}
\markup \justify-line {ooooooooo ooooooooo oo ooo}
```

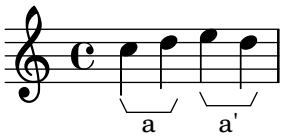
oooooo oooooo oooooo oooooo

ooooooooo oooooooo oo ooo

Editorial annotation improvements

- It is now possible to add text to analysis brackets through the `HorizontalBracketText` object.

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

{
  \once \override HorizontalBracketText.text = "a"
  c''\startGroup d''\stopGroup
  e''-\tweak HorizontalBracketText.text "a'" \startGroup d''\stopGroup
}
```



Text formatting improvements

- Support for making it easier to use alternative 'music' fonts other than the default Emmentaler in LilyPond has been added. See Section "Replacing the notation font" in *Notation Reference* for more information.
- Default text fonts have been changed from `Century Schoolbook L`, `sans-serif`, and `monospace`.

  For `svg` backend:

Family	**Default font**
roman	`serif`
sans	`sans-serif`
typewriter	`monospace`

  `serif`, `sans-serif`, and `monospace` are `generic-family` in SVG and CSS specifications.

  For other backends:

Family	**Default font (alias)**	**Alias definition lists**
roman	`LilyPond Serif`	TeX Gyre Schola, C059, Century SchoolBook URW, Century Schoolbook L, DejaVu Serif, ..., serif
sans	`LilyPond Sans Serif`	TeX Gyre Heros, Nimbus Sans, Nimbus Sans L, DejaVu Sans, ..., sans-serif
typewriter	`LilyPond Monospace`	TeX Gyre Cursor, Nimbus Mono PS, Nimbus Mono, Nimbus Mono L, DejaVu Sans Mono, ..., monospace

  `LilyPond Serif`, `LilyPond Sans Serif`, and `LilyPond Monospace` are font aliases defined in the LilyPond dedicated FontConfig configuration file `00-lilypond-fonts.conf`. Where a character dosen't exist in the first font listed, the next font listed will be used instead for that character. For details of alias definitions, please see to `00-lilypond-fonts.conf` under the installed directory.

- When using OpenType fonts, font features can be used. Note: Not all OpenType fonts have all functions.

```
% True small caps
\markup { Normal Style: Hello HELLO }
\markup { \caps { Small Caps: Hello } }
\markup { \override #'(font-features . ("smcp"))
          { True Small Caps: Hello } }

% Number styles
\markup { Normal Number Style: 0123456789 }
\markup { \override #'(font-features . ("onum"))
          { Old Number Style: 0123456789 } }

% Stylistic Alternates
\markup { \override #'(font-features . ("salt 0"))
          { Stylistic Alternates 0: εφπρθ } }
\markup { \override #'(font-features . ("salt 1"))
          { Stylistic Alternates 1: εφπρθ } }

% Multiple features
\markup { \override #'(font-features . ("onum" "smcp" "salt 1"))
          { Multiple features: Hello 0123456789 εφπρθ } }
```

Normal Style: Hello HELLO

Small Caps: Hello

True Small Caps: Hello

Normal Number Style: 0123456789

Old Number Style: 0123456789

Stylistic Alternates 0: εφπρθ

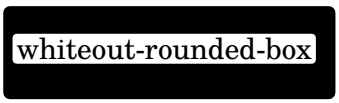
Stylistic Alternates 1: ϵϕϖϱϑ

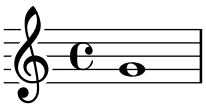
Multiple features: Hello 0123456789 ϵϕϖϱϑ

- Two new styles of whiteout are now available. The `outline` style approximates the contours of a glyph's outline, and its shape is produced from multiple displaced copies of the glyph. The `rounded-box` style produces a rounded rectangle shape. For all three styles, including the default `box` style, the whiteout shape's `thickness`, as a multiple of staff-line thickness, can be customized.

```
\markup {
  \combine
    \filled-box #'(-1 . 15) #'(-3 . 4) #1
    \override #'(thickness . 3)
    \whiteout whiteout-box
```

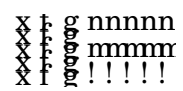
```
}
\markup {
  \combine
    \filled-box #'(-1 . 24) #'(-3 . 4) #1
    \override #'(style . rounded-box)
    \override #'(thickness . 3)
    \whiteout whiteout-rounded-box
}
\markup {
  \combine
    \filled-box #'(-1 . 18) #'(-3 . 4) #1
    \override #'(style . outline)
    \override #'(thickness . 3)
    \whiteout whiteout-outline
}
\relative {
  \override Staff.Clef.whiteout-style = #'outline
  \override Staff.Clef.whiteout = 3
  g'1
}
```





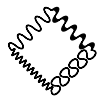
- A new markup-command, `\with-dimensions-from`, makes `\with-dimensions` easier to use by taking the new dimensions from a markup object, given as first argument.

```
\markup {
  \pattern #5 #Y #0 "x"
  \pattern #5 #Y #0 \with-dimensions-from "x" "f"
  \pattern #5 #Y #0 \with-dimensions-from "x" "g"
  \override #'(baseline-skip . 2)
  \column {
    \pattern #5 #X #0 "n"
    \pattern #5 #X #0 \with-dimensions-from "n" "m"
    \pattern #5 #X #0 \with-dimensions-from "n" "!"
  }
}
```



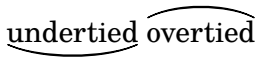
- Markup-command `\draw-squiggle-line` is now available. Customizing is possible with overrides of `thickness`, `angularity`, `height` and `orientation`

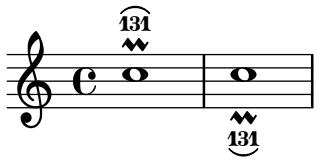
```
\markup
  \overlay {
    \draw-squiggle-line #0.5 #'(3 . 3) ##t

    \translate #'(3 . 3)
    \override #'(thickness . 4)
    \draw-squiggle-line #0.5 #'(3 . -3) ##t

    \translate #'(6 . 0)
    \override #'(angularity . -5)
    \draw-squiggle-line #0.5 #'(-3 . -3) ##t

    \translate #'(3 . -3)
    \override #'(angularity . 2)
    \override #'(height . 0.3)
    \override #'(orientation . -1)
    \draw-squiggle-line #0.2 #'(-3 . 3) ##t
  }
```



- Markup-commands `\undertie` and `\overtie` are now available, as well as the generic markup-command `\tie`.

```
\markup {
  \undertie "undertied"
  \overtie "overtied"
}

m = {
  c''1 \prall -\tweak text \markup \tie "131" -1
}

{ \voiceOne \m \voiceTwo \m }
```





New for specialist notation

Vocal music improvements

- A new flexible template suitable for a range of choral music, is now provided. This may be used to create simple choral music, with or without piano accompaniment, in two or four staves. Unlike other templates, this template is 'built-in', which means it does not need

to be copied and edited: instead it is simply `\include`'d in the input file. For details, see Section "Built-in templates" in *Learning Manual*.

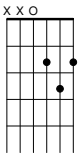
- The `\addlyrics` function now works with arbitrary contexts incuding `Staff`.
- `\lyricsto` and `\addLyrics` have been 'harmonized'. Both now accept the same kind of delimited argument list that `\lyrics` and `\chords` accept. Backward compatibility has been added so music identifiers (i.e. `\mus`) are permitted as arguments. A `convert-ly` rule has been added that removes redundant uses of `\lyricmode` and rearranges combinations with context starters such that `\lyricsto` in general is applied last (i.e. like `\lyricmode` would be).

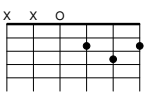
Unfretted and fretted string instrument improvements

- A new note head style for Tabulature has been added – `TabNoteHead.style = #'slash`.
- In fret-diagrams the distance between frets and the distance between strings is now independently adjustable. Available are `fret-distance` and `string-distance` as subproperties of `fret-diagram-details`.

```
fretMrkp = \markup { \fret-diagram-terse "x;x;o;2;3;2;" }

\markuplist
\override #'(padding . 2)
\table #'(0 -1) {
  "default"

  \fretMrkp

  "fret-distance"

  \override #'(fret-diagram-details . ((fret-distance . 2)))
  \fretMrkp

  "string-distance"

  \override #'(fret-diagram-details . ((string-distance . 2)))
  \fretMrkp
}
```

default 

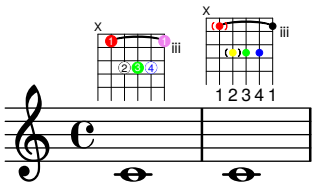
fret-distance 

string-distance 

- It is now possible to individually color both the dots and parentheses in fret diagrams when using the `\fret-diagram-verbose` markup command.

```
\new Voice {
```

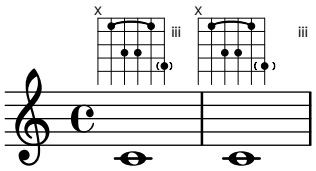
```
  c1^\markup {
    \override #'(fret-diagram-details . (
                 (finger-code . in-dot))) {
      \fret-diagram-verbose #'((mute 6)
                               (place-fret 5 3 1 red)
                               (place-fret 4 5 2 inverted)
                               (place-fret 3 5 3 green)
                               (place-fret 2 5 4 blue inverted)
                               (place-fret 1 3 1 violet)
                               (barre 5 1 3 ))
    }
  }
  c1^\markup {
    \override #'(fret-diagram-details . (
                 (finger-code . below-string))) {
      \fret-diagram-verbose #'((mute 6)
                              (place-fret 5 3 1 red parenthesized)
                              (place-fret 4 5 2 yellow
                                          default-paren-color
                                          parenthesized)
                              (place-fret 3 5 3 green)
                              (place-fret 2 5 4 blue )
                              (place-fret 1 3 1)
                              (barre 5 1 3))
    }
  }
}
```



- Two new properties have been added for use in `fret-diagram-details` when using the `\fret-diagram-verbose` markup command; `fret-label-horizontal-offset` which affects the `fret-label-indication` and `paren-padding` which controls the space between the dot and the parentheses surrounding it.

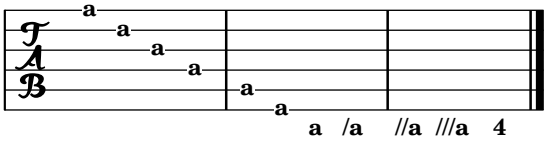
```
\new Voice {
  c1^\markup {
    \fret-diagram-verbose #'((mute 6)
                             (place-fret 5 3 1)
                             (place-fret 4 5 2)
                             (place-fret 3 5 3)
                             (place-fret 1 6 4 parenthesized)
                             (place-fret 2 3 1)
                             (barre 5 2 3))
  }
  c1^\markup {
    \override #'(fret-diagram-details . (
                 (fret-label-horizontal-offset . 2)
                 (paren-padding . 0.25))) {
```

```
      \fret-diagram-verbose #'((mute 6)
                               (place-fret 5 3 1)
                               (place-fret 4 5 2)
                               (place-fret 3 5 3)
                               (place-fret 1 6 4 parenthesized)
                               (place-fret 2 3 1)
                               (barre 5 2 3))
    }
  }
}
```



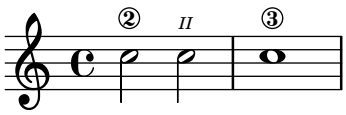
- Additional bass strings (for lute tablature) are supported.

```
m = { f'4 d' a f d a, g, fis, e, d, c,  \bar "|." }

\score {
  \new TabStaff \m
  \layout {
    \context {
      \Score
      tablatureFormat = #fret-letter-tablature-format
    }
    \context {
      \TabStaff
      stringTunings = \stringTuning <a, d f a d' f'>
      additionalBassStrings = \stringTuning <c, d, e, fis, g,>
      fretLabels = #'("a" "b" "r" "d" "e" "f" "g" "h" "i" "k")
    }
  }
}
```



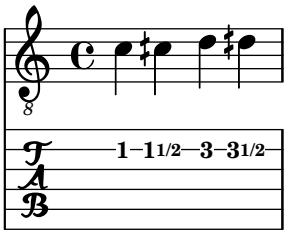
- String numbers can now also be used to print roman numerals (e.g. for unfretted string instruments).

```
c2\2
\romanStringNumbers
c\2
\arabicStringNumbers
c1\3
```



- TabStaff is now able to print micro-tones for bendings etc.

```
\layout {
  \context {
    \Score
    supportNonIntegerFret = ##t
  }
}

mus = \relative { c'4 cih d dih }

<<
  \new Staff << \clef "G_8" \mus >>
  \new TabStaff \mus
>>
```



Chord notation improvements

- \chordmode can now use < > and << >> constructs.
- It is now possible to override the text property of chord names.

```
<<
\new ChordNames \chordmode {
  a' b c:7
  \once \override ChordName.text = "foo"
  d
}
>>
```

A B C^{7} foo

New for input and output

Input structure improvements

- Blocks introduced with \header can be stored in variables and used as arguments to music and scheme functions and as the body of #{...#} constructs. They are represented as a Guile module.

  While \book, \bookpart, \score, \with, \layout, \midi, \paper blocks can be passed around in similar manner, they are represented by different data types.

Titles and header improvements

- Page numbers may now be printed in roman numerals, by setting the page-number-type paper variable.

Input file improvements

- A new command \tagGroup has now been added. This complements the existing \keepWithTag and \removeWithTag commands. For Example:

```
\tagGroup #'(violinI violinII viola cello)
```

declares a list of 'tags' that belong to a single 'tag group'.

```
\keepWithTag #'violinI
```

Is now only concerned with 'tags' from 'violinI''s tag group.

Any element of the included music tagged with one or more tags from the group, but *not* with *violinI*, will be removed.

Output improvements

- LilyPond source files may now be embedded inside the generated PDF files. This experimental feature is disabled by default and may be regarded as unsafe, as PDF documents with hidden content tend to present a security risk. Please note that not all PDF viewers have the ability to handle embedded documents (if not, the PDF output will appear normally and source files will remain invisible). This feature only works with the PDF backend.
- The `output-classic-framework` procedure and the `-dclip-systems` are now available with the `SVG` backend.
- An argument, `-dcrop`, has been added, formatting `SVG` and `PDF` output without margins or page-breaks.
- A new `output-attributes` grob property is now used for svg output instead of the `id` grob property. It allows multiple attributes to be defined as an association list. For example, `#'((id . 123) (class . foo) (data-whatever . \bar"))` will produce the following group tag in an SVG file: `<g id=\123" class=\foo" data-whatever=\bar"> ... </g>`.
- The PostScript functionality of stroke adjustment is no longer applied automatically but left to the discretion of the PostScript device (by default, Ghostscript uses it for resolutions up to 150dpi when generating raster images). When it is enabled, a more complex drawing algorithm designed to benefit from stroke adjustment is employed mostly for stems and bar lines.

  Stroke adjustment can be forced by specifying the command line option '`-dstrokeadjust`' to LilyPond. When generating `PDF` files, this will usually result in markedly better looking `PDF` previews but significantly larger file size. Print quality at high resolutions will be unaffected.
- Added a new `make-path-stencil` function that supports all `path` commands both relative and absolute:

  `lineto`, `rlineto`, `curveto`, `rcurveto`, `moveto`, `rmoveto`, `closepath`. The function also supports 'single-letter' syntax used in standard SVG path commands:

  `L`, `l`, `C`, `c`, `M`, `m`, `Z` and `z`. The new command is also backward-compatible with the original `make-connected-path-stencil` function. Also see `scm/stencil.scm`.

MIDI improvements

- The most common articulations are now reflected in MIDI output. Accent and marcato make notes louder; staccato, staccatissimo and portato make them shorter. Breath marks shorten the previous note.

  This behavior is customizable through the `midiLength` and `midiExtraVelocity` properties on `ArticulationEvent`. See `script-init.ly` for examples.
- Improved MIDI output for breathe marks. After tied notes, breaths take time *only* from the last note of the tie; e.g. `{ c4~ c8 \breathe }` performs as `{ c4~ c16 r }` instead of `{ c4 r8 }`. This is more consistent with articulations and how humans interpret breaths after ties. It now also makes it easier to align simultaneous breathe marks over multiple parts, all with different note lengths.
- There is now support for controlling the 'expression level' of MIDI channels using the `Staff.midiExpression` context property. This can be used to alter the perceived volume

of even sustained notes (albeit in a very 'low-level' way) and accepts a number value between `0.0` and `1.0`.

```
\score {
  \new Staff \with {
    midiExpression = #0.6
    midiInstrument = "clarinet"
  }
  <<
    { a'1~ a'1 }
    {
      \set Staff.midiExpression = #0.7 s4\f\<
      \set Staff.midiExpression = #0.8 s4
      \set Staff.midiExpression = #0.9 s4
      \set Staff.midiExpression = #1.0 s4

      \set Staff.midiExpression = #0.9 s4\>
      \set Staff.midiExpression = #0.8 s4
      \set Staff.midiExpression = #0.7 s4
      \set Staff.midiExpression = #0.6 s4\!
    }
  >>
  \midi { }
}
```

- When outputting MIDI, LilyPond will now store the `title` defined in a score's `\header` block (or, if there is no such definition on the `\score` level, the first such definition found in a `\header` block of the score's enclosing `\bookpart`, `\book`, or top-level scope) as the name of the MIDI sequence in the MIDI file. Optionally, the name of the MIDI sequence can be overridden using the new `midititle` `\header` field independently of `title` (for example, in case `title` contains markup code which does not render as plain text in a satisfactory way automatically).
- Support for making it easier to use alternative 'music' fonts other than the default Emmentaler in LilyPond has been added. See Section "Replacing the notation font" in *Notation Reference* for more information.

Extracting music improvements

- `\displayLilyMusic` and its underlying Scheme functions no longer omit redundant note durations. This makes it easier to reliably recognize and format standalone durations in expressions like

```
{ c4 d4 8 }
```

New for spacing issues

Page breaking improvements

- There are two new page breaking functions. `ly:one-page-breaking` automatically adjusts the height of the page to fit the music, so that everything fits on one page. `ly:one-line-auto-height-breaking` is like `ly:one-line-breaking`, placing the music on a single line and adjusting the page width accordingly, however it also automatically adjusts the page height to fit the music.

Vertical and Horizontal spacing improvements

- It is now possible to move systems with reference to their current position using the `extra-offset` subproperty of `NonMusicalPaperColumn.line-break-system-details`.

Both vertical and horizontal changes are possible. This feature is especially useful for making slight adjustments to the default vertical position of individual systems. See Section "Explicit staff and system positioning" in *Notation Reference* for more information.

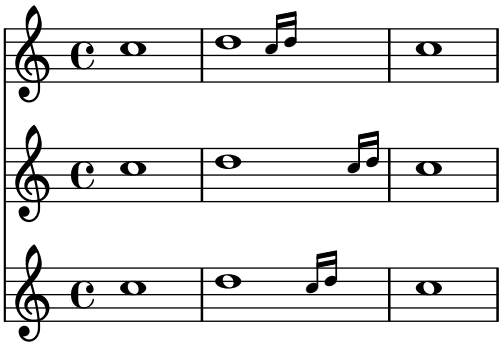
- Improved visual spacing of small and regular 'MI' Funk and Walker noteheads so they are now the same width as other shaped notes in their respective sets. `SOL` noteheads are also now visually improved when used with both the normal Aiken and Sacred Harp heads, as well as with the thin variants.
- `LeftEdge` now has a definable `Y-extent` (i.e.vertical). See Section "LeftEdge" in *Internals Reference*.
- Grobs and their parents can now be aligned separately allowing more flexibility for grob positions. For example the 'left' edge of a grob can now be aligned on the 'center' of its parent.
- Improved horizontal alignment when using `TextScript`, with `DynamicText` or `LyricText`.

New for changing defaults

An optional argument for the `\afterGrace` command has been added.

`\afterGrace` now has an optional argument to specificy the spacing fraction position of its notes.

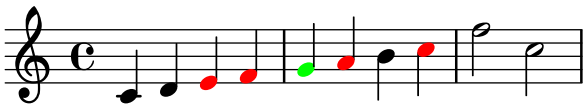
```
<<
\new Staff \relative {
  % The default, hard-coded value (3/4)
  c''1 \afterGrace d1 { c16[ d] } c1
}
\new Staff \relative {
  % Changing the hard-coded value manually (15/16)
  #(define afterGraceFraction (cons 15 16))
  c''1 \afterGrace d1 { c16[ d] } c1
}
\new Staff \relative {
  % Using the new argument (5/6)
  c''1 \afterGrace 5/6 d1 { c16[ d] } c1
}
>>
```



- All of `\override`, `\revert`, `\set`, and `\unset` now work with the `\once` prefix for making one-time settings.

```
\relative {
  c'4 d
  \override NoteHead.color = #red
  e4 f |
```

```
  \once \override NoteHead.color = #green
  g4 a
  \once \revert NoteHead.color
  b c |
  \revert NoteHead.color
  f2 c |
}
```

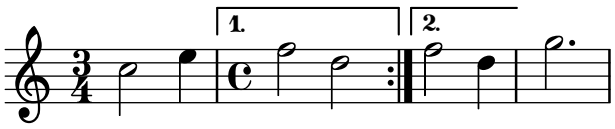


New for Internal interfaces and functions

- The music and grob property `spanner-id`, used for distinguishing simultaneous slurs and phrasing slurs, has been changed from a string to a *key* which can be either a non-negative integer or symbol (also see the previous documented expressive mark improvement – *A new command* \=*X has been added*).
- Context properties named in the '`alternativeRestores`' property are restored to their value at the start of the *first* alternative in all subsequent alternatives.

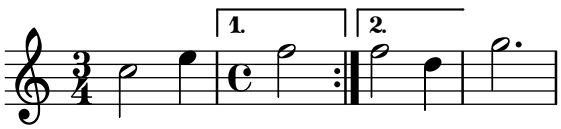
  Currently the default set restores 'current meter':

```
\time 3/4
\repeat volta 2 { c2 e4 | }
\alternative {
  { \time 4/4 f2 d | }
  { f2 d4 | }
}
g2. |
```



  'measure position':

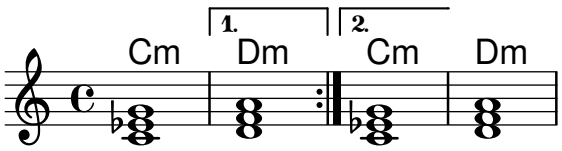
```
\time 3/4
\repeat volta 2 { c2 e4 | }
\alternative {
  { \time 4/4
    \set Timing.measurePosition = #(ly:make-moment -1/2)
    f2 | }
  { f2 d4 | }
}
g2. |
```



  and 'chord changes':

```
<<
  \new ChordNames {
```

```
    \set chordChanges = ##t
    \chordmode { c1:m d:m c:m d:m }
  }
  \new Staff {
    \repeat volta 2 { \chordmode { c1:m } }
    \alternative {
      { \chordmode { d:m } }
      { \chordmode { c:m } }
    }
  \chordmode { d:m }
}
>>
```



- LilyPond functions defined with `define-music-function`, `define-event-function`, `define-scheme-function` and `define-void-function` can now be directly called from Scheme as if they were genuine Scheme procedures. Argument checking and matching will still be performed in the same manner as when calling the function through LilyPond input. This includes the insertion of defaults for optional arguments not matching their predicates. Instead of using `\default` in the actual argument list for explicitly skipping a sequence of optional arguments, `*unspecified*` can be employed.
- Current input location and parser are now stored in GUILE fluids and can be referenced via the function calls `(*location*)` and `(*parser*)`. Consequently, a lot of functions previously taking an explicit `parser` argument no longer do so.

  Functions defined with `define-music-function`, `define-event-function`, `define-scheme-function` and `define-void-function` no longer use `parser` and `location` arguments.

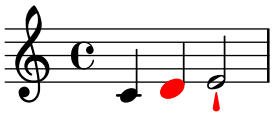
  With those particular definitions, LilyPond will try to recognize legacy use of `parser` and `location` arguments, providing backwards-compatible semantics for some time.
- Scheme functions and identifiers can now be used as output definitions.
- Scheme expressions can now be used as chord constituents.
- Music (and scheme and void) functions and markup commands that just supply the final parameters to a chain of overrides, music function and markup command calls can now be defined in the form of just writing the expression cut short with `\etc`.

```
\markup bold-red = \markup \bold \with-color #red \etc
highlight = \tweak font-size 3 \tweak color #red \etc

\markup \bold-red "text"
\markuplist \column-lines \bold-red { One Two }

{ c' \highlight d' e'2-\highlight -! }
```

text

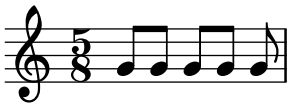
One

Two



- Dot-separated symbol lists like `FretBoard.stencil` were already supported as of version 2.18. They may now also contain unsigned integers, and may alternatively be separated by commata. This allows usage such as

```
{ \time 2,2,1 5/8  g'8 8 8 8 8 }
```

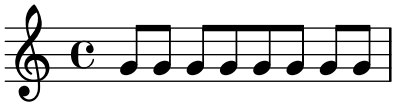


and

```
\tagGroup violin,oboe,bassoon
```

- Such lists may also be used in expressions for assignments, sets, and overrides. This allows usage such as

```
{ \unset Timing.beamExceptions
  \set Timing.beatStructure = 1,2,1
  g'8 8 8 8 8 8 8 8 }
```



- Association list elements could previously be assigned values individually (for example, paper variables like `system-system-spacing.basic-distance`). They may now be also referenced in this manner, as with

```
\paper {
  \void \displayScheme \system-system-spacing.basic-distance
}
```

In combination with the previously mentioned changes, this allows setting and referencing pseudovariables like `violin.1`.

- The markup-list-command `\table` is now available. Each column may be aligned differently.

```
\markuplist {
    \override #'(padding . 2)
    \table
      #'(0 1 0 -1)
      {
        \underline { center-aligned right-aligned center-aligned left-aligned }
        one "1" thousandth "0.001"
        eleven "11" hundredth "0.01"
        twenty "20" tenth "0.1"
        thousand "1000" one "1.0"
      }
}
```

<u>center-aligned</u> <u>right-aligned</u> <u>center-aligned</u> <u>left-aligned</u>

one	1	thousandth	0.001
eleven	11	hundredth	0.01
twenty	20	tenth	0.1
thousand	1000	one	1.0

- `InstrumentName` now supports `text-interface`.
- The `thin-kern` property of the `BarLine` grob has been renamed to `segno-kern`.
- `KeyCancellation` grobs now ignore cue clefs (like `KeySignature` grobs do).
- Add support for `\once \unset`