

Funcionalidades nuevas de la versión 2.16 desde la 2.14

- Se han simplificado las instrucciones de los sellos de elementos gráficos para permitir una menor duplicación de código y mejores aproximaciones de altura de los objetos gráficos. Se han eliminado las siguientes instrucciones de sello:
 - beam
 - bezier-sandwich
 - bracket
 - dashed-slur
 - dot
 - oval
 - repeat-slash
 - zigzag-line
- Ahora se tratan los corchetes como objetos diferenciados y no como parte de la plica.



- Se puede elegir entre dos métodos de numeración de compases, en especial para cuando se emplean repeticiones:

Six musical staves in treble clef with a common time signature 'C'. The first staff shows a sequence of notes with a bracket above the last two measures labeled '1.'. The second staff shows a sequence of notes with a bracket above the last two measures labeled '2.'. The third staff shows a sequence of notes with a bracket above the last two measures labeled '3.'. The fourth staff shows a sequence of notes with a bracket above the last two measures labeled '1.'. The fifth staff shows a sequence of notes with a bracket above the last two measures labeled '2.'. The sixth staff shows a sequence of notes with a bracket above the last two measures labeled '3.'. The staves are labeled 1, 2, 3, 5, 6b, and 6c respectively on the left side.

- Lo que sigue es un cambio fundamental en la representación que LilyPond hace de la música: los eventos de duración como `LyricEvent` y `NoteEvent` ya no se encuentran envueltos dentro de elementos `EventChord` a no ser que se hayan escrito realmente como parte de un acorde. Si manipulamos expresiones musicales en Scheme, el nuevo comportamiento puede necesitar cambios en nuestro código. Las llamadas a la función musical `\eventChords` o a la función de Scheme `event-chord-wrap!` convierten a la representación anterior; la utilización de una cualquiera de ellas puede ser la vía más sencilla para mantener operativo el código tradicional.

Las ventajas de hacer que la entrada y la música tengan una más estrecha correspondencia son numerosas: las funciones musicales funcionaban anteriormente de forma distinta si se utilizaban dentro o fuera de los acordes. Ahora son lo mismo, incluidas todas las posibilidades del análisis sintáctico de los argumentos. Ahora podemos usar variables musicales dentro de los acordes: una construcción como

```
tonic=fis'
{ <\tonic \transpose c g \tonic> }
```



habría sido impensable con anterioridad. Podemos usar `#{...#}` para construir los componentes de un acorde. Las funciones musicales dentro de los acordes ya no se manejan de ninguna forma especial y por tanto aceptan los mismos argumentos que fuera de los acordes. La instrucción `\tweak` funciona ahora sobre notas individuales sin necesidad de envolverlas entre ángulos de acorde. En teoría, también puede funcionar sobre eventos y sobre la letra de las canciones. Dado que antes no era posible, depende de la suerte caso por caso si las interioridades del código de los trucos están recibiendo ya la información necesaria. Se solicita a los usuarios que informen de aquellos casos en que se observe que la instrucción `\tweak` no funciona según lo que razonablemente se espera de ella.

- Como consecuencia, era posible reimplementar la abreviatura de repetición de acordes `q`. Los acordes repetidos ahora se sustituyen justo antes de interpretar la expresión musical. En caso de que el usuario quiera retener ciertos eventos del acorde original, puede ejecutar manualmente la función de sustitución de repetición de acordes `\chordRepeats`.
- Las expresiones de Scheme dentro de fragmentos de código de LilyPond incrustados (`#{...#}`) se ejecutan ahora dentro de la cerradura léxica del código de Scheme circundante. El símbolo `$` ya no es especial dentro del código de LilyPond incrustado. Se puede utilizar de forma incondicional dentro de código de LilyPond para su evaluación inmediata, de forma parecida a la forma en que se utilizaba anteriormente `ly:export`. Se ha suprimido `ly:export`. Como consecuencia, ahora `#` está libre para diferir la evaluación de su argumento hasta que el analizador sintáctico reduzca efectivamente la expresión contenida, reduciendo significativamente el potencial de la evaluación prematura.
- Se ha mejorado el soporte de acordes de tipo jazz: se reconocen los acordes lidios y alterados; ahora se tratan los separadores entre modificadores de acorde de forma independiente de los separadores entre acordes invertidos y sus notas de bajo (y por omisión, la barra inclinada se usa ahora solamente para el último tipo de separador); las notas adicionales ya no van prefijadas por "add" de forma predeterminada; y la "m" en los acordes menores se puede personalizar. Consulte [Sección “Nombres de acorde personalizados” in Referencia de la Notación](#) para más información.
- Se ha cambiado el nombre de la instrucción `\markuplines` por `\markuplist` para conseguir una mejor correspondencia con su semántica y con la nomenclatura general de LilyPond.

- Se ha simplificado considerablemente la interfaz para especificar afinaciones en las tablaturas y se emplea la función de Scheme `\stringTuning` para la mayor parte de los propósitos.
- Las barras ahora pueden preservar la inclinación por encima de los saltos de línea.



Para hacerlo, se han hecho obsoletas varias funciones de "callback".

- `ly:beam::calc-least-squares-positions`
- `ly:beam::slope-damping`
- `ly:beam::shift-region-to-valid`

Además, `ly:beam::quanting` ahora acepta un argumento adicional para ayudar a los cálculos sobre los cambios de línea. Todas estas funciones se llaman automáticamente cuando se ajusta el parámetro `positions`.

- En los argumentos de función, la música, los elementos de marcado y las expresiones de Scheme (así como algunas otras entidades sintácticas) se han hecho mayormente intercambiables y se diferencian solamente mediante la evaluación del predicado respectivo. En ciertos casos, el analizador sintáctico consulta este predicado, como cuando se decide si interpretar `-3` como un número o como un evento de digitación.
- Ahora se pueden definir las funciones musicales (y sus parientes cercanos) con argumentos opcionales.
- Para definir instrucciones que se ejecutan solamente por sus efectos secundarios, ahora está disponible `define-void-function`.
- Hay una instrucción nueva `define-event-function` en analogía con `define-music-function` que se puede usar para definir funciones musicales que actúan como post-eventos sin que se requiera un especificador de dirección como `(-, ^ o _)` antes de ellos.

```
dyn=#(define-event-function (parser location arg) (markup?)
      (make-dynamic-script arg))
\relative c' { c\dyn pfsss }
```



- Se puede incluir una lista de alias en ASCII para caracteres especiales.

```
\paper {
  #(include-special-characters)
}
\markup "&bull; &dagger; &copyright; &OE; &ss; &para;"
```

• † © Œ ß ¶

- Hay una instrucción nueva `define-scheme-function` en analogía con `define-music-function` que puede usarse para definir funciones que se evalúan a expresiones de Scheme pero aceptan argumentos en la sintaxis de LilyPond.
- Ahora se puede utilizar la construcción `#{ ... #}` no solo para crear listas secuenciales de música, sino también para alturas (que se distinguen de los eventos de nota sencillos por la ausencia de duración u otra información que no puede formar parte de una altura), eventos musicales únicos, expresiones musicales vacías, post-eventos, elementos de marcado (sobre todo para liberar a los usuarios de la necesidad de usar la macro `markup`), listas de marcado, expresiones numéricas, definiciones y modificaciones de contextos y algunas otras cosas. Si no contiene nada o contiene un único evento musical, ya no devuelve una lista secuencial de música, sino una expresión musical vacía o simplemente el propio evento musical, respectivamente.
- Se pueden usar alturas en la parte derecha de las asignaciones. Las alturas se diferencian de los eventos de una sola nota en que no tienen duración ni otras informaciones que no pueden formar parte de una altura.
- Nueva opción de la línea de órdenes `--loglevel=level` para controlar el volumen de datos que LilyPond produce en la salida. Los valores posibles son ERROR (errores), WARN (advertencias), BASIC_PROGRESS (progreso básico), PROGRESS (progreso) y DEBUG (depuración).
- `\once \set` ahora reinicia correctamente el valor de la propiedad al valor previo.



- La alineación de los elementos de matiz dinámico extensos (reguladores, crescendo textuales, etc.) se divide automáticamente si se da explícitamente una dirección distinta.



- Ahora las apoyaturas y mordentes funcionan también dentro de una ligadura de expresión, y no solo dentro de una ligadura de fraseo. Asimismo, se ha añadido la función `\slashedGrace` que no imprime ninguna ligadura partiendo de la nota del mordente.



- Para suprimir a línea en un elemento de crescendo extenso (y otros elementos extensos similares), LilyPond contempla ahora de forma plena la propiedad `#'style = #'none`.



- LilyPond.app está disponible ahora para MacOS X 10.7. ¡Gracias, Christian Hitz!
- Los glissandos pueden abarcar varias líneas.