Creating new symbols in abcm2ps

Hudson Lacerda (2004−2005)

English version by Eduardo Vieira da Silva (2006)

(Re)definition of the PostScript operators

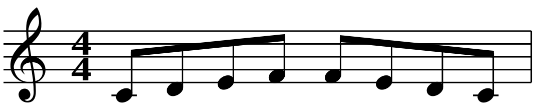
The score generator `abcm2ps` enables the creation and modification of *PostScript* operators. This is a very useful resource for customizing scores, as well as introducing nonexistent symbols in the program.

In order to send a line of *PostScript* to the output, you can simply use **`%%postscript`** in the file ABC.

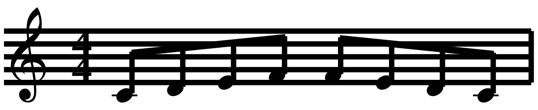
The example below changes the thickness of the staff line by redefining the operator **`dlw`** generated by `abcm2ps`:

```
%%postscript /dlw {2.5 setlinewidth} def
```

original dlw (0.7pt)



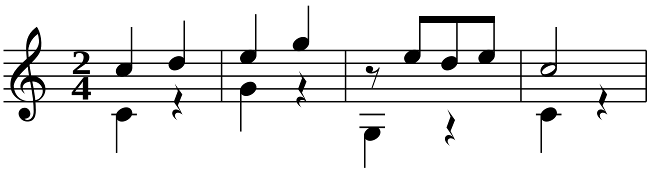
redefined dlw (2.5pt)



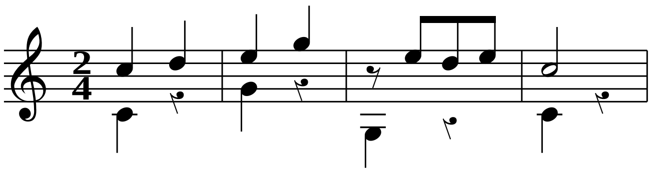
The following example redefine the operator **`r4`** − which draws the fourth note rest by using the eighth note rest as a base (**`r8`**).

```
%%postscript /r4{gsave translate [−1 0 0 1 0 0] concat 0 0 r8 grestore}def
```

original r4



redefined r4 (mirrored eighth note)

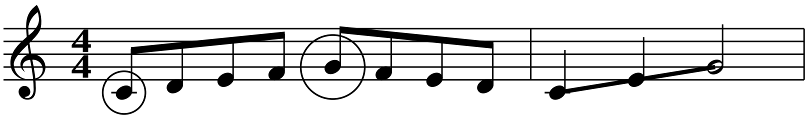


A *PostScript* code can be very useful for inserting drawings in a very accurate position in relation to the note. The example below enables the writing of commands in an annotation (*annotation*). The annotation text must begin with the character **`':'`** so that it may be interpreted as a command

```
%%postscript /cp {currentpoint}!
%%postscript /anshow { dup 0 get (:) 0 get ne
%%postscript     {/gchshow load cshow}
%%postscript     {dup length 1 sub 1 exch getinterval cvx exec}ifelse}!
```

By including the above code in the beginning of anABC file, you can run commands like the following ones:

```
X:1
T:Commands in annotations
K:C
"@0,0:cp newpath 10 360 0 arcn stroke"CDEF \
"@0,0:cp newpath 15 360 0 arcn stroke"GFED |\
"@0,0:cp /yy exch def /xx exch def"C2 E2 \
"@0,0:gsave xx yy lineto 2 setlinewidth stroke grestore"G4|
```



New symbols (%%deco)

Besides the possibilities that were already shown in the given examples, `abcm2ps` allows the addition of new decorations (symbols such as accents, dynamic, articulations, etc.). This is done through the command **`%%deco`**. The syntax is the following:

`%%deco <name> <type> <ps> <height> <widthL> <widthR> [text]`

`<name>` is the symbol identifier;

`<type>` is a number form 0 to 7 which defines the type of symbol;

`<ps>` is the name of the *PostScript* operator to be run;

`<height>` is the vertical space taken by the symbol in points;

`<widthL>` and **`<widthR>`** indicates the left and right space of the symbol (Since `abcm2ps-4.8.2`, it has only worked for type 6 decorations);

`[text]` is a (optional) text to be used by the *PostScript* operator (used only by type 3, 4 and 6 decorations).

The **`<type>`** of the symbols defines its positioning

`0:` close to the note head, like **`!tenuto!`** or **`.`** (*staccato*);
`2:` on the left of a note head, like **`!slide!`**;
`2:` on the left of a chord, like **`!arpeggio!`**;
`3, 4:` generic expressions above and below the staff;
`5:` long symbol above the staff, like **`!trill(!`** or **`!trill)!`**;
`6:` generic (usually below the staff);
`7:` long symbol below the staff, like **`!crescendo(!`** or **`!crescendo)!`**.

Symbols of the 0 through 2 type are associated with the notes, and are placed within the staff. Types 3 to 5 are placed outside the staff, although they are associated with the notes. In turn, the types 6 and 7 are associated to the staff.

There is steel another special category of decorations: when its **`<name>`** begins with **`head-`**, the decoration replaces the note head in which it is used.

How to create a type 0 decoration

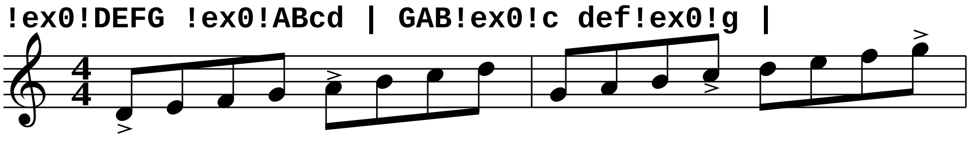
Type 0 decorations are placed close to the note head Two examples are **`!tenuto!`** and **`!dot!`**. The *PostScript* operator must use two arguments: the coordinates **`<x>`** and **`<y>`**, which determine the position of the symbol The command **`%%deco`** must not make use of the **`[text]`** argument.

Here is an example (close to the note head):

```
%%postscript /example0 {  moveto
%%postscript              -3 -2 rmoveto
%%postscript               6  2 rlineto
%%postscript              -6  2 rlineto
%%postscript              stroke
%%postscript           } def
```

```
%%deco ex0 0 example0 5 0 0
```

```
X:1
K:C
!ex0!DEFG !ex0!ABcd | GAB!ex0!c def!ex0!g |
```



How to create a type 1 decoration

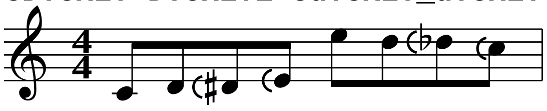
Type 1 decorations are placed on the left of the note head An example is **!slide!**. Just like the procedure for type 0, *PostScript* operator must use two arguments: the coordinates a**<x>** and **<y>**, which determine the position of the symbol The command **%%deco** must not make use of the **[text]** argument. The argument **<height>** is irrelevant, which can simply be zero. Notice that the position of the symbol is affected by the presence of an accidental.

Here is an example (half moon on the left of the note):

```
%%postscript /example1 {  newpath
%%postscript              exch 1 add exch
%%postscript              5 100 260 arc
%%postscript              1.2 setlinewidth stroke dlw
%%postscript            } def
```

```
%%deco ex1 1 example1 0 0 0
```

```
X:1
K:C
CD!ex1!^D!ex1!E ed!ex1!_d!ex1!c |
```



How to create a type 2 decoration

Type 2 decorations are placed on the left of chord and their height spans the whole extent of the chord. An example is **!arpeggio!**.

The *PostScript* operator must use three argument: the minimum height of the symbol and vertical and horizontal coordinates of the lowest symbol.

The command **%%deco** must not make use of the **[text]** argument The The **<height>** height is used to define the minimal height of the symbol (since abcm2ps−4.8.8).

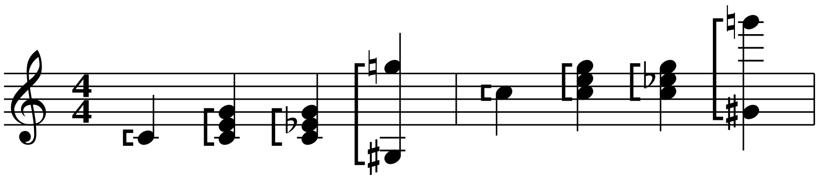
The positioning of the symbol is affected by the presence of accidentals.

Here is an example (brace):

```
%%postscript /exemplo2 {   moveto
%%postscript               4 0 rmoveto
%%postscript              -4 0 rlineto 0 exch rlineto
%%postscript               4 0 rlineto 1.6 setlinewidth stroke dlw
%%postscript            } def
```

```
%%deco ex2 2 exemplo2 6 0 0
```

```
X:1
L:1/4
K:C
!ex2!C !ex2![CEG] !ex2![C_EG] !ex2![^G,=g] |\
!ex2!c !ex2![ceg] !ex2![c_eg] !ex2![^G=g'] |
```



How to create a type 3 decoration

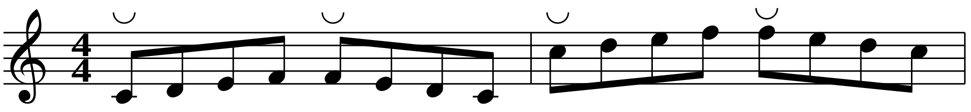
Type 3 decorations are usually placed above the staff. Some examples are: **`!accent!`**, **`!segno!`** e **`!mordent!`**. The *PostScript* operator must use two or three arguments: the horizontal and vertical coordinates and, optionally, a text *string* (in case **`%%deco`** command should use the argument **`[text]`**).

Here is an example (not−accent):

```
%%postscript /exemplo3a {  newpath
%%postscript               7.5 add
%%postscript               5 180 360 arc
%%postscript               stroke
%%postscript            } def
```

```
%%deco ex3a 3 exemplo3a 10 0 0
```

```
X:1
K:C
!ex3a!CDEF !ex3a!FEDC | !ex3a!cdef !ex3a!fedc |
```

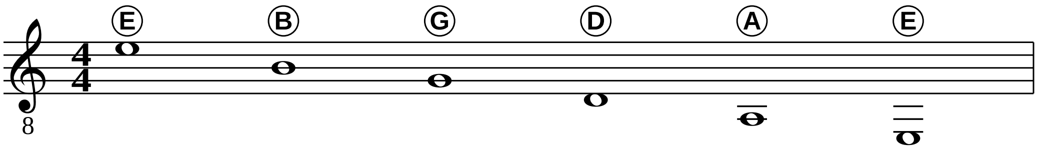


Now an example with text (string indications):

```
%%postscript /exemplo3b {  4 add 2 copy
%%postscript               newpath 4 add 7 0 360 arc stroke
%%postscript               moveto
%%postscript               /Helvetica−Bold 12 selectfont
%%postscript               showc
%%postscript            } def
```

```
%%deco ex3b_E 3 exemplo3b 16 0 0 E
%%deco ex3b_B 3 exemplo3b 16 0 0 B
%%deco ex3b_G 3 exemplo3b 16 0 0 G
%%deco ex3b_D 3 exemplo3b 16 0 0 D
%%deco ex3b_A 3 exemplo3b 16 0 0 A
```

```
X:1
L:1/1
K:C treble−8
!ex3b_E!e !ex3b_B!B !ex3b_G!G !ex3b_D!D !ex3b_A!A, !ex3b_E!E, |
```



How to create a type 4 decoration

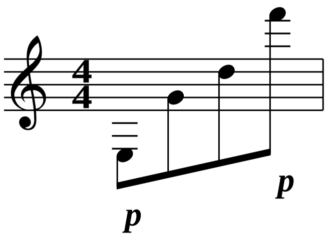
Type 3 decorations are usually placed below the staff.

Just like the procedure for types 3 and 6, the *PostScript* operator must use two or three arguments: the horizontal and vertical coordinates and, optionally, a text *string* (in case, **`%%deco`** command should use a the argument **`[text]`**).

An example of its utilization is the creation of symbols for dynamics that are close to the notes (instead of being horizontally aligned). The **`pf`**operator from `abcm2ps` (fit for printing dynamics symbols) is used in the following example. (See also ‘‘How to create a type 6 decoration.)

```
%%deco ex4 4 pf 20 0 0 p

X:1
K:C treble
!ex4!E,Gd!ex4!f' |
```



How to create a type 6 decoration

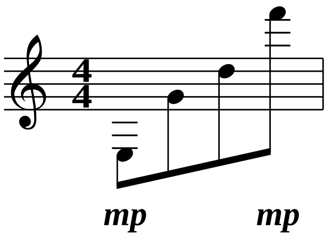
Type 3 decorations are usually placed below the staff. Typical examples are the dynamics symbols, such as: **`!p!`**, **`!mf!`** and **`!ff!`**.

Just like the procedure for types 3 and 4, the *PostScript* operator must use two or three arguments: the horizontal and vertical coordinates and, optionally, a text *string* (in case, **`%%deco`** command should use the argument **`[text]`**).

A simple and useful example is the implementation of the dynamics *mp*. The **`pf`** operator from `abcm2ps` (fit for printing dynamics symbols) is used in the following example. Notice that the type 6 decorations, unlike type 4, are horizontally aligned

```
%%deco ex6 6 pf 20 10 8 mp

X:1
K:C treble
!ex6!E,Gd!ex6!f' |
```



How to create a type 5 decoration

Type 5 decorations are long signs placed above the staff. An example is the long trill starting with **`!trill(!`** and ending with **`!trill)!`**.

In the decoration which starts the symbol, the field **`<ps>`** (*PostScript* operator) of the 3%%deco, must be **`'-'`**. The **`<name>`** of the initial decoration typically ends with **`'('`**, where as the name of the final decoration ends with **`')'`**.

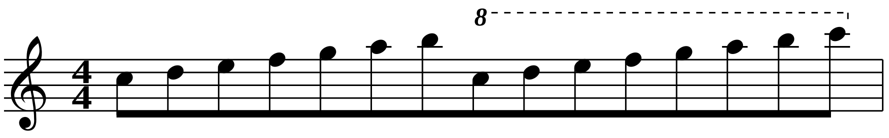
The *PostScript* operator of the final decoration is the one in charge of the drawing, and receives three arguments: the span of the symbol and the horizontal and vertical coordinates of the first part of the symbol.

Here is an example (Ottava brackets):

```
%%postscript /exemplo5 {
%%postscript              moveto -3 3 rmoveto
%%postscript              /Times-BoldItalic 12 selectfont (8) show
%%postscript              gsave
%%postscript                 2 6 rmoveto
%%postscript                 [3] 0 setdash 12 add 0 rlineto
%%postscript                 currentpoint stroke
%%postscript                 moveto [] 0 setdash 0 -3 rlineto stroke
%%postscript              grestore
%%postscript           } def
```

```
%%deco ex5( 5 - 10 0 0
%%deco ex5) 5 exemplo5 20 0 0
```

```
X:1
K:C
cdefgab!ex5(!cdefgab!ex5)!c' |
```



How to create a type 7 decoration

Type 7 decorations are long signs placed below the staff, such as the dynamics symbols **!crescendo(!**, **!crescendo)!**, **!diminuendo(!** and **!diminuendo)!**.

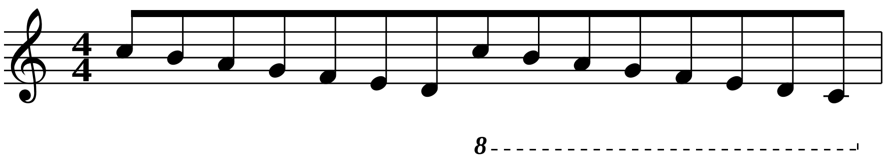
They are created in the same way as the type 5 decorations, except that the type 7 decorations are printed below the staff.

Here is an example (Ottava bassa):

```
%%postscript /exemplo7 {
%%postscript              moveto -6 3 rmoveto
%%postscript              /Times-BoldItalic 12 selectfont (8) show
%%postscript              gsave
%%postscript                 2 2 rmoveto
%%postscript                 [3] 0 setdash 12 add 0 rlineto
%%postscript                 currentpoint stroke
%%postscript                 moveto [] 0 setdash 0 3 rlineto stroke
%%postscript              grestore
%%postscript           } def
```

```
%%deco ex7( 7 - 10 0 0
%%deco ex7) 7 exemplo7 20 0 0
```

```
X:1
K:C
cBAGFED!ex7(!cBAGFED!ex7)!C |
```



How to create a head−xxx (note head) decoration

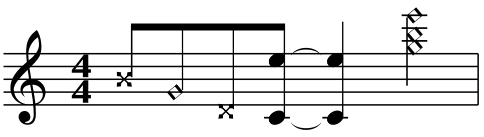
A decoration whose **<name>** begins with **head-** replaces the note head in which it is used. **head-xxx** decorations are declared as being of any type (from 0 through 7), using the proper parameters for the specified type.

Two useful examples are the diamond shaped note heads (harmonic) and in the shape of **x** (percussion). In this example, the **x** is actually a *double sharp* (drawn by the *PostScript* operator **dsh0**, generated by abcm2ps). Notice also the effect of a **head−xxx** decoration in a chord: it is applied to all the notes of the chord.

```
%%postscript /harm{gsave T 40 rotate
%%postscript            .25 dup setlinewidth
%%postscript            -3 2.3 3 -1 roll 2 div sub M 6 0 RL
%%postscript            0 -4.6 RM                   -6 0 RL
%%postscript            stroke
%%postscript            1.4 dup setlinewidth
%%postscript            3 1 index 2 div sub 2.3 M 0 -4.6 RL
%%postscript            -6 add 0 RM               0  4.6 RL
%%postscript            stroke
%%postscript      grestore}!
```

```
%%deco head-h 0 harm 0 0 0
%%deco head-x 0 dsh0 0 0 0
```

```
X:1
K:C
!head-x!B!head-h!G!head-x!D[Ce]-[Ce]2 !head-h![gbe']2 |
```



Decorations in single notes in chords

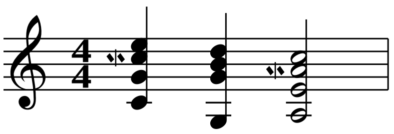
A decoration can be applied to a single note of a chord (*i.e.* a note inside **'['** and **']'**). In this case, a regular decoration could be misplaced. Therefore you should resort to a special *PostScript* operator (and a special decoration) in order to use it in a single note of a chord.

As an example, a mordent (based on the **lmrd** operator of abcm2ps) which is positioned on the left of the note head The symbol was declared as a type 1 decoration in order to gain a little bit of space on the left.

```
%%postscript /morde{exch 11 sub exch 4 sub lmrd}!
```

```
%%deco morde 1 morde 0 0 0
```

```
X:1
K:C
[CG!morde!ce]2 [G,GBd]2 [A,E!morde!Ac]4 |
```



Microtonal accidentals

With abcm2ps it is possible to have special accidentals for microtonalism. Micro tonal pitches are indicated by a fraction after an accidental, like **^3/4c**. By *default*, the numerator is set to 1, and the denominator to 2 (**^/c** is the same as **^1/2c**). The numerator and denominator must be whole numbers from one to 256.

1/2 and 3/2 sharps or flats are supported (for quarter−tones). For other values, new *PostScript* operators must be created (with **%%postscript**). The name of a *PostScript* operator for an accidental must be:

`<accidental_type><micro_value>`

where **`<accidental_type>`** will be: **`sh`** (sharp), **`ft`** (flat), **`nt`** (natural), **`dsh`** (double sharp) or **`dft`** (double flat).

`<micro_value>` is a number calculated from the fraction as:

`(numerator−1)*256 + (denominator−1)`.

In the following example, the first line of music employs the accidentals which are available in `abcm2ps`. The second line employs alternative accidentals: a combination of a sharp or a flat with an arrow In the `ABC` code for this example, an accidental with **`9/`** (or **`9/2`**) uses an up arrow, and an accidental with **`6/`** (or **`6/2`**) uses a down arrow.

Notice the value of **`<micro_value>`** within the name of the *PostScript* operators, calculated for the fractions **`9/2`** and **`6/2`**:

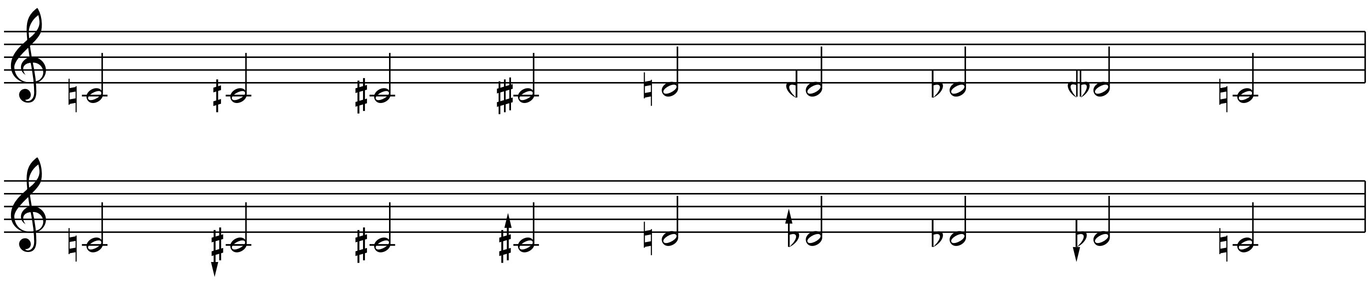
`(9−1)*256 + (2−1) = 2049`

`(6−1)*256 + (2−1) = 1281`

Here is the code:

```
%%postscript /seta {M 1.7 −7 RL −3.4 0 RL fill}!
%%postscript /sh2049 {2 copy sh0 exch 1.5 add exch 15 add seta}! % ^9/2
%%postscript /sh1281 {2 copy sh0 exch 1.2 sub exch 15 sub        % ^6/2
%%postscript          gsave T 180 rotate 0 0 seta grestore}!
%%postscript /ft2049 {2 copy ft0 exch 1.8 sub exch 14 add seta}! % _9/2
%%postscript /ft1281 {2 copy ft0 exch 1.8 sub exch 11 sub        % _6/2
%%postscript          gsave T 180 rotate 0 0 seta grestore}!
```

```
X:1
M:none
L:1/2
K:C
=C  ^/C ^C ^3/C =D  _/D _D _3/D =C |
=C ^6/C ^C ^9/C =D _9/D _D _6/D =C |
```



Some special operators

defl

The operator **`defl`** (introduced with `abcm2ps−4.8.0`) is a flag, which indicates:

− the state of a type 5 or 7 decoration (beginning or ending);

− the direction of the note tail (up or down).

This operator is generated/refreshed for each note containing the decoration

In order to use it, from `abcm2ps−4.9.4` on, it is necessary to set the **`%%setdefl command to true`**.

The first bit to the right **`defl`** indicates if a long decoration is starting (**`0`**) or if it is a continuation (**`1`**) (refers to the left end of the decoration);

The second bit, from the right indicates if the long decoration is ending (**0**) or if it will be continued on the next line (**1**) (refers to right end of the decoration);

The third bit from the right indicates if the tail is up (**0**) or down (**1**). It is useful to create decorations that will change depending on the direction of the tail (for example, one that will be drawn right on tail itself).

So, in order to check the state of the first bit (on the left end of a long decoration) you can use a sequence of *PostScript* with the following structure:

```
defl 1 and 0 eq {
    % procedure in case the left end is the beginning of the decoration
}{
    % % procedure in case the left end is the continuation of the decoration
} ifelse
```

In order to check if the second bit (right end of a long decoration):

```
defl 2 and 0 eq {
    % procedure in case the right end is the end of the decoration
}{
    % procedure in case the right end is the continuation of the decoration
} ifelse
```

And in order to check the third bit (tail direction):

```
defl 4 and 0 eq {
    % procedure in case the tail is down
}{
    % procedure in case the tail is up
} ifelse
```

Examples:

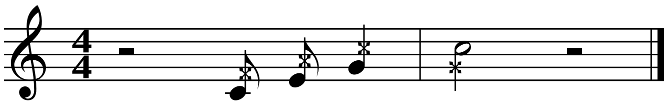
The example below uses **defl** to determine where the tail is so that a **x** (*sprechgesang*) can be drawn on it.

The decoration is declared is being one of type 1, because its position is always relative to the note head. As a side effect, abcm2ps leaves an extra space on the left of the note (where it would usually be drawn) such type of decoration). In case you don’t want this extra space, the you can declare a type 0 (or 3) and always use such decoration (together with the note) between square bracket, as in a chord.

```
%%setdefl true
%%postscript /sprgsg{gsave T
%%postscript         defl 4 and 0 eq{1.5 -9}{8.5 9}ifelse T
%%postscript         .8 dup scale 0 0 dsh0 grestore}!

%%deco sprgsg 1 sprgsg 0 0 0

X:1
K:C
z4 !sprgsg!C !sprgsg!E !sprgsg!G2 | !sprgsg!c4 z4 |]
```



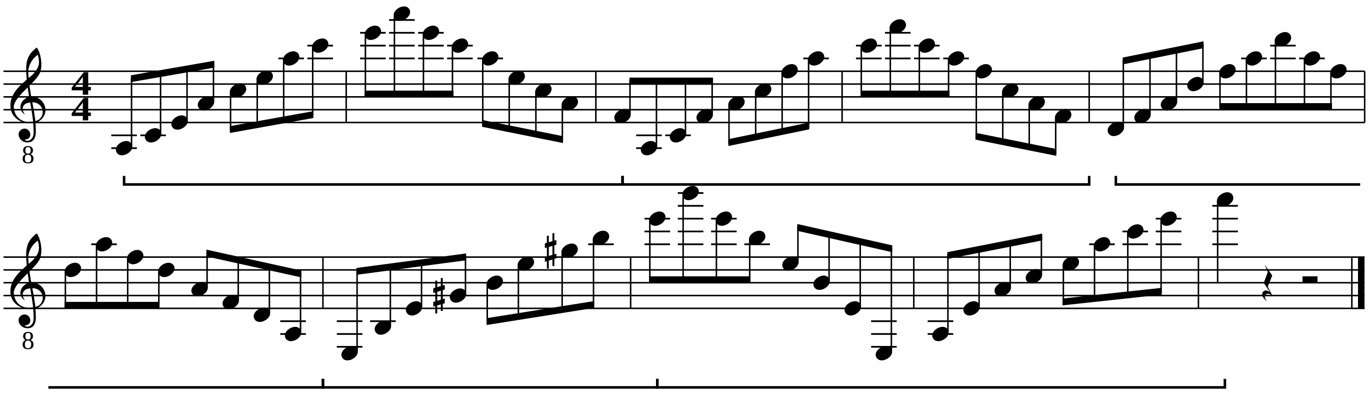
In the following example an bracket below the staff is implemented (which can be used as indication for piano pedal). The beginning and the end of this long decoration are detected thanks to the **defl** operator. Notice (in the score) that the bracket that starts in the last measure of the of the first line remains ‘open’, as well as the left side of its continuation on the next line.

The fist line of *PostScript* code defines the **defl** operator with zeros. This is totally unnecessary (and has no effect) with abcm2ps−4.8.0 or more recent, but makes the code compatible with earlier versions. When you use an earlier version of the program, the decoration will always be be drawn with a beginning and an end.

```
%%setdefl 1
%%postscript /defl 0 def
%%postscript /bracket { exch 3 sub exch 1.2 setlinewidth
%%postscript             defl 1 and 0 eq{4 add M 0 −4 RL}{M}ifelse
%%postscript             7 add 0 RL
%%postscript             defl 2 and 0 eq{0 4 RL}if
%%postscript             stroke dlw}!

%%deco bracket( 7 − 12 0 0
%%deco bracket) 7 bracket 12 0 0

X:1
K:Am clef=treble−8
%
!bracket(! A,CEA ceac’  |  e’a’e’c’ aecA  |\
!bracket)! !bracket(! FA,CF Acfa  |  c’f’c’a fcAF  !bracket)! |\
!bracket(! DFAd fad’af  |
%
dafd AFDA,  !bracket)! !bracket(! |\
E,B,E^G Be^gb  |  !bracket)! !bracket(! e’b’e’b eBEE,  |\
A,EAc eac’e’  |  !bracket)! a’2 z2 z4  |]
```



The same principles could be applied to improve the examples of the type 5 and type 7 decorations given earlier (lines for octave above and below).

Changing the note heads globally

In some cases it is desirable to change globally the note heads in order to ‘stylize’ the score or to create special resources, for example. In order to accomplish this, you can just redefine the *PostScript* operators which are in charge of drawing the note heads.

The operators in `abcm2ps` that draw note heads are:

hd	Black head, for fourth notes and shorter notes
Hd	Half note
HD	Whole note
HDD	Breve (rounded)
breve	Breve (square)
longa	Longa
pshhd	Used in a note with a sharp, when the clef is **perc** (abcm2ps−4.4.5)
pflhd	Used in a note with a flat, when the clef is **perc** (abcm2ps−4.4.5)
ghd	Black head for grace notes (ornaments)

All of these note heads store their position through the **x** and **y** operators. They are used to position other symbols, like tails and augmentation dots

This implies that, in order to write other versions for the note heads, it is imperative to define **x** e **y**. There are at least two ways of accomplishing this:

(1) Use the operator **xymove**, which moves to the (*x*,*y*) point and saves its coordinates.

(2) Define **x** and **y** ‘manually’, in case you don’t want to move the (*x*,*y*) point. This can be accomplished with:

```
/y exch def /x exch def
```

(which takes the two values on top of the stack and saves them as **x** as **y**);

or with:

```
/x 2 index def /y 1 index def
```

(which takes the two values on top of the stack *without deleting them*).

Here is an example of redefining the note head in order to draw a melodic contour:

```
%%postscript /hdo/hd load bind def     % stores a copy of the original ‘hd’
%%postscript /xx 10000 def /yy 0 def   % initialize ‘previous note’
%%postscript /hd { xymove              % links to the previous note (left)
%%postscript       x xx gt {xx yy M x y lineto stroke} if
%%postscript       /xx x def /yy y def}!

X:1
L:1/8
M:none
K:C clef=perc stafflines=1  % (stafflines were introduced in abcm2ps−4.8.6)
[M:4/4] AFgG, ad’be’ | [M:7/16] c’’/a/e/b/ C/B/A/ | [M:3/8] E,3 |
%%postscript /hd/hdo load bind def     % restores original ‘hd’
```