

An fvwm primer

Victor Eijkhout

February 2002

1 Who? What? Why?

The `fvwm2` window manager (I am using `fvwm` here but I really mean its latest version, `fvwm2`) is not the most sexy looking, but it is lean and mean. Its man pages are intended more as a reference than a tutorial, and I was not able to find any ‘gentle’ introductions to `fvwm`. Hence this tutorial. I will not explain every last command and option, just enough to get you set up. After that you can read the man pages yourself.

Comments and suggestions are always welcome by email.

2 Setup files

You need a file `.xinitrc` that starts up X programs, and a file `.fvwm2rc` that describes how windows are organised. A minimal `.xinitrc` file would have

```
xterm -geometry 80x40+100+100 &  
exec /sw/bin/fvwm2
```

or whatever the path to `fvwm2` is. If it is in your search path, just use `exec fvwm2`.

Another file you may want to have is `.Xdefault`, which sets up default values for X applications. For instance, you could set the default geometry of xterms (see the example above) in that file. Lines in `.Xdefaults` look like

```
xterm*geometry: 80x24
```

Just as in the shell, lines in `.fvwm2rc` starting with a `#` character are ignored. Quotes are only necessary to group words separated by a space

There is a good discussion of what to put in the background and the use of `exec` on <http://fink.sourceforge.net/doc/x11/run-xfree86.php>; see section 4.4. For a discussion of various startup files, see http://www.perchine.com/dyp/x/online/X_startup.html.

3 The pager

With `fvwm` you can have multiple virtual desktops. The tool to navigate between these is called the ‘pager’. You declare a pager by having a statement

```
Module FvwmPager first last
```

in your `.fvwmrc` file, where `first last` describes how which virtual desktops are visible. If `first>0`, you can actually create windows that are not on any of the desktops in the pager. The layout in rows and columns of the pager is determined by

```
*FvwmPagerRows 2
*FvwmPagerColumns 2
```

It is convenient if your pager is visible on all desktops and does not get buried under other windows.

```
Style "FvwmPager" Sticky,StaysOnTop
You specify the style of the pager with commands like
Style "Fvwm Pager" Color red/green
See section 5.1 for more about styles.
```

To label your desktops, use commands like

```
*FvwmPagerLabel 0 Junk
*FvwmPagerLabel 1 Mail
```

4 Virtual desktops

As described in section 3, you can have several desktops. Each desktop in the pager can be larger than the physical screen. You declare the desktopsize by

```
DeskTopSize mxn
```

where `m` is the number of columns and `n` the number of rows. For the placement of windows on the screens and desktops, see section 5.4.

If you run the cursor into the side of your screen, it can slide into the next screen on this desktop, or stay on the current one. Similarly, you can move windows from one screen to the next by dragging them. By

```
EdgeResistance scrolling moving
```

you indicate how easy or hard such movement is. The first parameter is the number of milliseconds the cursor has to spend at the edge of the screen before it is moved to the next desktop, the second parameter is in pixels. With

```
EdgeScroll horizontal vertical
```

you indicate by what percentage of a desktop the view jumps.

```
EdgeResistance 400 0
EdgeScroll 100 100
```

You can only scroll between the windows of a virtual desktop, not between desktops themselves.

Navigation between windows and desktops is further explained in section 8.

5 Window attributes

5.1 Window styles

You specify window styles by

```
Style windowname options
```

for instance

```
Style "*" Title, Color red/black
```

The window name can be the actual name, for instance given by `xterm -title "my title"` or a class, or a resource string; see section 5.2.

Styles can be specified cumulatively: after

```
Style * Title
```

```
Style "Fvwm Pager" NoTitle
```

the pager has no title.

5.2 Style definitions

5.3 Window positioning on the screen

The location and size of a window are set in the `.xinitrc` file, for instance

```
xclock -geometry 130x120-5+5 &
```

The `mxn` specification is for the size. It seems that the size of `xterms` is measured in characters, for other programs it is in pixels. The `+m+n` specification determines the location horizontally and vertically on the screen, where a positive number is the offset from top/left and negative from bottom/right.

If you omit the location and the placement modes in the style, the window will have to be placed interactively when the window manager starts up.

5.4 Window positioning on the desktop

By default, windows will open on the top left screen of the first desktop; see section 4 for an explanation of screens and desktops.

To make a window visible on all screens and desktops, you have to declare it 'sticky'. For the pager (section 3) this is done with

```
Style "FvwmPager" Sticky,StaysOnTop
```

For named programs, you can use

```
Style "xclock" Sticky
```

and for named windows you can use

```
Style "console" NoTitle,Color blue/gray,Sticky
```

where the name was created with the `title` option in the `.xinitrc` file:

```
xterm -C -geometry 85x10+0-0 -title "console" &
```

5.5 Title bar buttons

Windows can have buttons in their title bars. The presence of such buttons is determined by the declaration of mouse actions for them:

```
Mouse button context modifier function
```

where the `button` is the number of the mouse button, and giving the context a digit value makes it a title bar button. Context values of 1 3 5 7 9 place buttons from the left inward, values 0 8 6 4 2 place buttons from the right inward.

You can bind a single command to a button, as in section 7, but more usefully, you can bind popup menus. See section 6.

6 Popup menus

You can make a menu appear by the command

```
PopUp menuname
```

or

```
Menu menuname double-click-action
```

for instance activated by a mouse click:

```
Mouse 0 1 N PopUp "window ops"
```

Menus activated by Menu are sticky; the ones activated with PopUp require you to hold the mouse down. The double-click action in Menu can often be taken as Nop.

The actual menu is created by a call to AddToMenu and subsequent extension lines:

```
AddToMenu "window ops"  
+ "Close" Close  
+ "Iconify" Iconify
```

Another useful pair of items to have in a menu somewhere is

```
+ "Refresh" Refresh  
+ "Source fvwmrc" Read /home/joebob/.fvwm2rc
```

The first one redraws all windows, the second one rereads the settings file.

The problem with rereading the settings file is that your decors, menus and functions are created with AddTo . . . commands. Therefore, you should have corresponding DestroyMenu and other statements before the first AddTo . . . line. You should also kill modules, for instance writing

```
KillModule FvwmPager  
Module FvwmPager 0 4
```

To give a menu a title and underline it, use the Nop operator:

```
AddToMenu "window ops"  
+ "Window stuff" Nop  
+ " " Nop  
+ "Close" Close  
+ "Iconify" Iconify
```

or just put the title on the AddToMenu line, as illustrated next. The Nop mechanism can be used for separators.

Submenus are created by having another popup inside a menu:

```
+ "Exit fvwm" Popup "verify quit"  
AddToMenu "verify quit" "Really quit?" Title  
+ "Yes. Really." Quit  
+ "Nonono. My mistake" Nop
```

7 Iconifying

The command Iconify toggles the iconification state of a window. You can iconify a window any number of ways, for instance by clicking on a title bar button:

```
Mouse 0 3 N Iconify
```

In order to de-iconify it, you need to set the mouse context to I:

```
Mouse 0 I N Iconify
```

If your iconified windows lands on top of the pager, and maybe other places too, it may be hard to de-iconify again. You can try to make the icons land in a certain region of the screen by

```
Style xterm IconBox 0 80 -0 -0
```

where the four coordinates are left, top, right, bottom. In this example, the top 80 rows of pixels of the screen are avoided.

Plain de-iconifying will leave the cursor in the place where the icon used to be. To move it, do something like

```
AddToFunc De-Iconify
+ "I" Iconify -1
+ "I" WindowId $w WarpToWindow 30 30
Mouse 0 I N De-Iconify
```

8 Navigation

You can navigate between the screens on a desktop with

```
Scroll hor ver
```

where the offsets are in percentage of screen size.

The Scroll command can be bound to keys using the

```
Key key context modifier command
```

For instance, here is how to setup control-arrowkey navigation:

```
Key Right A C Scroll 100 0
Key Left A C Scroll -100 0
Key Up A C Scroll 0 -100
Key Down A C Scroll 0 100
```

Navigating between desktops uses the Desk command. For example:

```
Key Right A SC Desk 1
Key Left A SC Desk -1
```

move forward and backward one desk. To move to a specific desk, use

```
Desk 0 nn
```

Find key names in `/usr/include/X11/keysymdef.h` with `XK_` removed.

9 Programs and customisations

You can start programs from fvwm by the Exec command. For instance,

```
AddToMenu Applications "Applications" Title
+ "xterm" Exec rxvt -geometry 80x40
+ "emacs" Exec emacs -geometry 80x40
Mouse 0 A C Menu "Applications" Nop
```

creates a menu (see section 6) with title Applications that pops up by a mouse control-click in any application window, and the menu options are to start a terminal or to start emacs.

9.1 Xterm

The command to install an xterm is, surprise, `xterm`. However, `rxvt` is memory friendlier, and it seems its scroll bars are better behaved. Speaking of scroll bars, `xterm` doesn't have any by default; use the `-sb` option.

The number of saved lines in `rxvt` is by default a bit low; put

```
rxvt*saveLines: 1200
```

(or something like that) in your `.Xdefaults` file.

9.2 Emacs

Emacs has for some bizarre reason bound its control-H key to the help function. Put

```
(define-key global-map "\C-h" 'backward-delete-char)
```

into your `.emacs` file.