Jo's alternative desktop: PWM

CLEVERLY CONTRIVED

How would you like a window manager with support for dockapps, which let you access several applications from tabs on a single window? Jo Moskalewski takes investigates PWM, which does just that

On his Web site, *http://www.students.tut.fi/~tuomov/pwm/*, PWM's author Tuomo Valkonen declares that his window manager "may not be the easiest window manager to get into, but most good things aren't". In fact the list of features absent from PWM may quickly convince many a Linux Magazine reader that it would be better to look elsewhere. However, it's also clear that the program's failings are soon more than compensated for by its unusual functions.

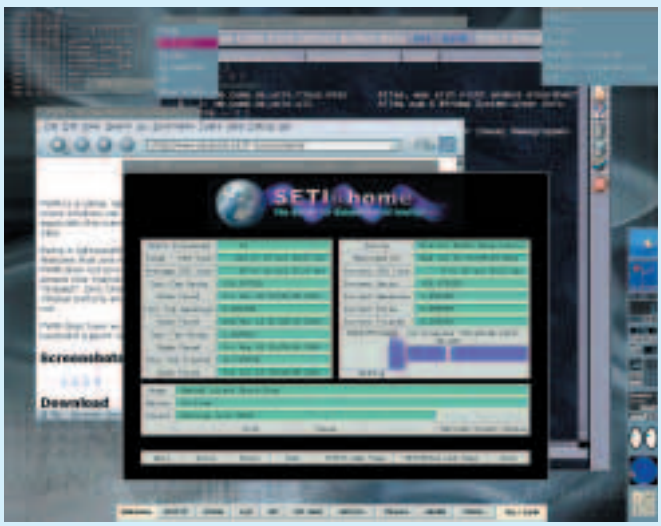
Faulty goods

Before we get down to PWM's juicy features, let's pick out the bones on which some users might choke.

The biggest difference between the various window managers lies in their focus behaviour and program-specific handling. PWM provides only the so-called "sloppy focus": with this, the window over which you place your mouse responds to keyboard inputs. However, it continues to be covered up by other windows until you bring it into the foreground manually. You can do this either by a mouse click or via a keyboard command.

Such behaviour is unfortunately something you'll just have to put up with, as is PWM's appearance. Its window decorations light up only in simple, plain colours. Anyone who wants to use colour shadings, or even graphics, will have no luck with this window manager. Another peculiarity is the lack of any windows buttons, with which windows can be closed or maximised. Instead of these, a right mouse click on the title bar activates a drop-down menu with these functions. However, if you're expecting to see an "iconify" option here then you'll be disappointed as it's just not present. To complete your misery, PWM doesn't come with any configuration tool of its own, but leaves this task to your favourite text editor. Many of you may of course regard that as a plus: it is after all very nice to be able to use your usual editor, instead of having to burrow through a new GUI.

There is yet more delight when one notices that the windows may not be able to reduce to an icon, but instead to their small titlebar. Double-clicking on this bar with the left mouse button very practically "rolls" the application up.



Main PWM screen

deskTOPia

Only you can decide how your Linux desktop looks. With deskTOPia we regularly take you with us on a journey into the land of window managers and desktop environments, presenting the useful and the colourful, viewers and pretty toys.

Compensation

One of the most outstanding characteristics of this desktop controller is hidden, not behind the left, but the middle mouse button: with this you can rip down a title bar (Figure 1) and drag it to another window bar. After this action PWM displays the applications concerned in the same window. The title bar then mutates – as can be seen in Figure 2 – into tabs. Again using the middle mouse button, it is now possible to select the desired application. A newly added program is thereby adapted to the size of the existing window. It is downright astonishing how easily this can be used to place and use any number of applications on a single desktop.

Figure 1: Title bar being moved

Figure 3: Dockapps

Damned nuisance

PWM also has a number of **virtual desktops**: while one can be devoted to the actual work, another one can be used for Web browsing and the next for image processing. If ever space on the desktop runs out, then you just take an empty new one.

If you want already-opened windows are to be dragged along when you switch desktops, then you simply pin them to the surface of the screen: a right mouse click on the title bar opens the window menu which also contains the entry "Tg stick". If a window has activated this option, a marking in the top right hand corner of the title bar indicates this. An application thus marked is present on all virtual desktops. If you select the corresponding menu item again, the sticker is removed and the application is left behind in the current workspace.

Menus can also be pinned on – not to take them with you on your journeys through various workspaces, but to keep them open at all times in the visual field. A mouse click on a menu title bar is a stepping-stone to continuous display, and a double click (or the Esc key) lays the ghost to rest again. PWM has two menus: the "Go to window" menu accessed via the middle mouse button (task list), which toggles between the available windows, and a start menu accessed via the right mouse button.

Into the bargain

Anyone who hankers for more than just these features should take a deep breath: PWM has a dock for Windowmaker "dockapps". These are special programs, reduced to 64x64 pixels, which act as icons and are designed to be clipped (docked) onto the window manager Windowmaker's interface. PWM users need no longer even flick them on: their dock automatically draws dockapps to the right spot, and does so reliably, as Figure 3 shows.

Figure 2: Three Konquerors and one ATerm in a single PWM window

The dock always remains in the foreground – any overlapping active window pushes itself underneath. If it ever really gets in your way, you can simply minimise it with the key combination Mod1+T. On most keyboards and system configurations Mod1 corresponds to the Alt key.

It's almost superfluous to mention that PWM can also be used fully and sensibly via the keyboard. The standard assignment is shown in Table 1.

On a plate

You will soon have this window manager installed: either play in a ready-made rpm package with the distribution package manager, or else go for the source. In the latter case the requirements of your computer are gleefully meagre – you only need the XFree86 developer package (usually called *xdevel* or similar) and the tool *make* together with compiler. After that the source code can be installed as follows:

```
tar xvfz pwm-1.0.tar.gz
cd pwm-1.0
make depend && make
su
cd pwm-1.0
make install
```

Starting PWM is somewhat more intricate than installation. Apart from a few distribution-dependent variants, however, the following path should lead to the destination: the crux of this is the file *~/.xinitrc* (or for a graphical login *~/.xsession*). If this file does not exist on your computer it can simply be made from new. Its content could look like this:

```
#!/bin/sh
exec pwm
```

Make the whole file executable, and next time you invoke *startx* (or at the next graphical login) PWM will greet you – even if this happens in a highly inconspicuous way.

Getting cosy

If the gentle PWM user does not make his or her own configuration file, the window manager uses the system one under */etc/pwm/pwm.conf* (or */usr/local/etc/pwm/pwm.conf*). The user's own desires can be turned into reality in the newly made file *~/.pwm/pwm.conf*. You can certainly bundle the entire configuration into a single *~/.pwm/pwm.conf*, but it is still possible to distribute the content among a number of configuration files. These must then be invoked from *pwm.conf*.

It's certainly not advisable to alter the system configuration files (with the exception of the start menu) – incorrect entries completely deactivate their function. Anything which is not validly configured, will

later not even be available. If, for example, you do not define any mouse operation, then you won't be able to use the mouse! For this reason it is better simply to convert your own desires at user level and otherwise to leave the system configurations as they are.

An example of a personal configuration file can be found at */etc/pwm/sample.conf*, which you can simply copy to *~/.pwm/pwm.conf*. This is the central configuration file, into which the start menu, the keyboard assignment and the mouse operation are loaded first (at which point you can type their content, if you like, in its entirety into this one file; the only thing to suffer will be clarity):

```
include "menus-default.conf"
include "keys-default.conf"
include "buttons-default.conf"
```

If you would like to set up a few defaults at this point, then simply make such a file according to the examples from */etc/pwm* and enter the modified files into your *pwm.conf*. The files to be loaded in can be found in both the system, as well as in the user's own directory – PWM searches both, but prefers the user directory *~/.pwm*. If your alterations are now to take effect, then it helps to restart the window manager: click with the right mouse button on the clear desktop, then select *Exit* followed by *Restart*.

Treat for the eyes

The example configuration file continues with the appearance:

```
screen 0 {
 include "look-brownsteel.conf"
 workspaces 6
 dock "-0-0", 1
}
```

As well as look-brownsteel.conf, look-beoslike.conf also comes with the package (additional "look-*.conf" files can be found on the coverdisc). A look into one of these files reveals that it is easy to create your own colour settings.

Six virtual desktops (*Workspaces*) should be enough for anyone, but if you like you can of course increase this number. Finally, the dock is placed in the bottom right-hand corner (geometry data "-0-0") and horizontally aligned. Anyone who prefers it vertical, should set, instead of the final "1" a "0".

In this section you can also make further entries: so with the line *font "lucida"*, in future the font *Lucida* will be used. *opaque_move 50* means that a window with a size of more than 50 per cent of the desktop area is shifted so that only the window frame, but not the content, is displayed (a feature which owners of decrepit old computers should value). The full list of possible tweaks for the screen section (and other sections) can be found in the file */usr/*/doc/pwm/config.txt*.

Table 1: Keyboard combinations

Command	Function
Alt+Tab	Change active window including raising
Alt+(1-9,0)	Change workspace ("0" is workspace 10)
Alt+M	Open start menu
Alt+G	"Go to window" menu – the task list
Alt+D	"Detach" menu – release attached window
Alt+T	Maximise or minimise dock
Alt+E	Call up new *xterm*
Alt+Return	The active window can then be moved by arrow keys
Shift+Ctrl+W	Close application and close window
Shift+Ctrl+X	Close window, without closing application (application crashes!)
Shift+Ctrl+S	Reduce window to title bar
Shift+Ctrl+Z	Pin on window
Shift+Ctrl+V	Maximise window vertically
Shift+Ctrl+H	Maximise window horizontally
Shift+Ctrl+M	Maximise window
Shift+Ctrl+(R/L)	Raising
Shift+Ctrl+A	Pin window to another
Shift+Ctrl+O	Window menu

Magic box

It gets really interesting with the following lines:

```
winprop "Netscape.Navigator" {
 frame 10
}
```

This makes pop-up (or mischievously newly opened) Navigator windows appear automatically in one and the same PWM window. Opera's properties can thus be combined with Netscape Navigator, although this is not really the intention at all.

With PWM it's not only possible to combine windows – if required, one can also show applications without window frames. For example to start the system monitor xosview without a frame, the following entry helps:

```
winprop "*.xosview" {
 wildmode yes
}
```

Conversely, with *no* instead of *yes* a window frame can be forced, if, at its own initiative, an application wants to go without it.

Because of its docks and its unique window handling options, PWM is definitely an odd sort of program. Its fundamentally different concept should be regarded as an opportunity for a different way of doing things, rather than as a failing. Anyone who takes a good look into this one may very well in future look at the greats of this genre with a bit of sympathy.

Virtual desktop: Most window managers offer several "screens", which can be filled with windows or applications. You can switch between these without having to close an application, but you can only see those applications that were started on the current desktop.