4

Overview

The window manager is responsible for managing application windows and communicating with the X server. The window manager controls the placement, borders, and decorations of any windows. It is responsible for the look and feel of your particular X session. This chapter introduces you to window manager functionality and to some of the many window managers found in today's Linux world.

This chapter also introduces the X desktop environments, which provide a modern GUI desktop, with integrated applications and a common programming environment. As we'll see, the desktop environment isn't an alternative to a window manager. Instead, it works with a window manager to provide a total working environment. The discussion in this chapter is necessarily general. Details are given in the individual window manager and desktop environment chapters.

Window Manager Features

The job of a window manager is to manage windows containing application output on the screen, as well as to handle input from the pointer and keyboard. The first window managers did that and not much else. Today's window managers still provide those capabilities but are now much more complicated programs and are customizable in just about any way imaginable. They now all have sorts of fancy options that let your desktop look unique (and certainly better and more pleasing!) compared to anyone else's desktop.

The various types of functionality are explained in the following sections. The Linux window managers mentioned later in this chapter and those described in detail in the following chapters support most of these functionalities using the pointer buttons as described. However, you should be aware that each window manager decides which features to support and how to support each. In addition, the pointer-button bindings are often configurable, which means that you as the window manager's user can strongly influence the window manager's capabilities.

Window Border

When you start an application, it usually comes up on your screen with a border surrounding it, as can be seen in Figure 4-1. This border, also known as the window's decoration, contains a title bar, several control buttons, and resize handles. The border is the primary interface to controlling an application window.

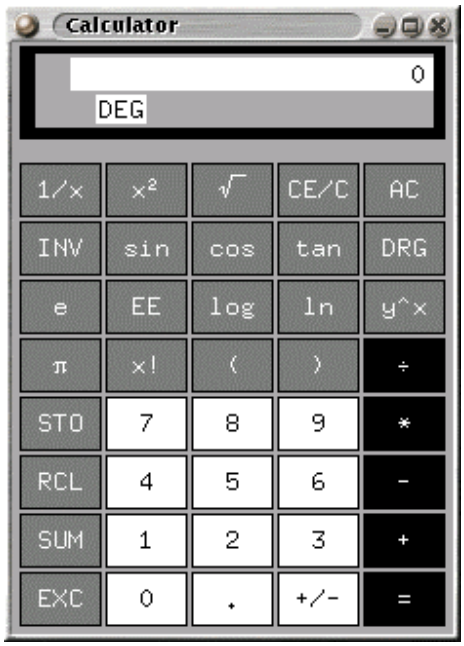

Figure 4-1: Window border

[Illustration department - Please add labels for "Window menu button" (top-left corner), "Title bar", "Minimize button", "Maximize button", and "Close button" to this figure. Please also add "Client area" pointing to somewhere in the middle and "resize bar" pointing to the window's border.]

Menu button
: The menu button opens a menu allowing you to choose operations such as move, resize, close or other operations to the window.

Title bar
: The title bar presents the name of the application, document or window. Its color indicates the keyboard focus state and type of window. You use the title bar to move, activate, raise, or lower the window.

Minimize button
: Click on the minimize button with the left pointer button to minimize or miniaturize (iconify) the window. This button is therefore sometimes referred to as the miniaturize button.

Maximize button
: Clicking on the maximize button increases the size of the window to completely fill the screen.

Close button
: The close button is used to close the window or kill the application.

Resize bar
: The resize bar is used to resize a window. The longer section in the middle of the resize bar only allows vertical resizing, while the corner sections, if available, allow the resizing of the window in both directions.

Client area
: The client area is where the application displays its information.

The window border style can vary strongly among the window managers presented in the second half of this chapter. The window manager's visual influence is purely subjective. However, its choice of buttons and resize handles is an objective influence on its window handling possibilities.

Using the Window Border

We've just seen that what looks like a simple border around the window actually provides the tools for managing our windows. In this section, we'll see how to use those tools.

Using the window operations menu

Clicking the right pointer button on the window's title bar or any button on the window's operations menu button often displays a menu that has window operations listed in it. Usually the possible operations include move, resize, minimize, maximize, and close the window. Because the details and choices of this menu differ from one window manager to another, it is described individually for each window manager in the following several chapters.

Moving a window

To move a window, click on the title bar of the window you want to move with the first pointer button and drag it with the button pressed. This also brings the window up to the front and focuses it.

Resizing a window

Dragging the resize bar changes the size of a window. Depending on where you grab the resize bar, the resize action is constrained to one or two dimensions:

- To change the window's height, grab the resize bar in the middle and drag it vertically.
- To change the window's width, grab the resize bar in either of its end regions and drag it horizontally.
- To simultaneously change the window's height and width, grab the resize bar in either of its end regions and drag it diagonally. Not all window managers support this option.

Minimizing a window

To minimize (iconify) a window, click on its minimization button. The window shrinks into an icon with a title bar, usually placed at the bottom of the screen. You can move this

icon around the screen by dragging it with the pointer while holding down the first pointer button. Examples of icons can be seen later in Figure 4-3.

To restore an icon back to its original window, double-click on the icon. The window is restored to the same position, size, and contents as it had before it was minimized. Actually, its contents might have changed if processing still went on while the window was iconified, thus possibly altering its contents.

Maximizing a window

If you want to resize a window to occupy the whole screen, it's easier to maximize it with a single pointer click instead of manually resizing it using its resize bars. When you later unmaximize it, the window is restored to the size and position it was before you maximized it.

To maximize a window, click on its maximization button. While most window managers only support a maximization to the full size of the screen, some window managers only maximize the window to the full height or width of the screen, depending on which pointer button is clicked. To restore a maximized window to its original size, click on this button again.

Shading a window

Shading a window is a function that not every window manager provides. If you want to temporarily have a window occupy less space on your screen you can shade it. This term comes from rolling up the shade of a real, glass window. When you roll up an X window you only see its title bar as shown in Figure 4-2. You can still operate on the title bar as if it were a regular window: move, minimize, or even close it.

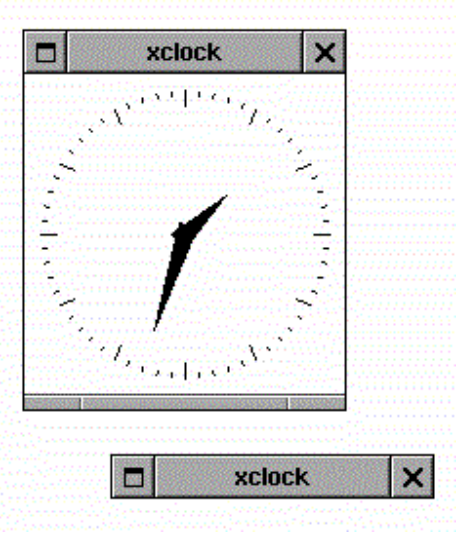

Figure 4-2: A normal window and its shaded version

To shade a window, double-click on the window's title bar. To unshade a window, double-click on its shade.

Closing a window

After finishing work in a window, you can close it to completely get rid of it. It is then removed from the screen and can no longer be restored. It is strongly suggested that you first properly save the window using the application's menus or buttons so as not to lose any information.

To close a window, click on the close button with the left pointer button.

Window Focus Policies

Windows can be either focused or unfocused. The focused window (also called the key or active window) usually has a title bar or window border that makes it stand out compared to the other windows. Unfocused windows have a less dominant title bar or window border.

As partially already seen in Chapter 2, *Getting Started Using X*, there are three styles of window focusing:

Manual focus mode, sometimes referred to as click-to-focus or explicit mode
: In manual focus mode, you explicitly select the window you want focused by placing the pointer in it and then clicking the first pointer button.

Auto-focus mode, also know as pointer, focus-follows-pointer or real-estate-driven mode
: In this mode, the focused mode is chosen based on the position of the pointer. The window underneath the pointer is always the focused window. This includes the root window.

Semi-auto-focus mode, commonly referred to as sloppy-focus mode
: This is similar to the auto-focus mode, but if you move the pointer from a non-root window to the root window, the original window does not lose its focus.

  Not all window managers support this mode.

When a window receives focus it not only gets a dominant frame, it is also raised to the top.

Reordering overlapping windows

Windows can overlap other windows, putting some windows over or in front of others. To use a window optimally, you need to both focus it and move it to the front. Actually as we've just seen, focusing a window usually brings it up front.

Icons and Icon Boxes

An icon is a small symbol that represents a full-sized window that is hidden by the window manager. While a window is hidden (or "iconified"), all processes that were happening in the window continue, but you cannot enter input or read output. Often window managers include an icon manager or icon box, as seen in Figure 4-3, to manage the many, often small, icons that accumulate.

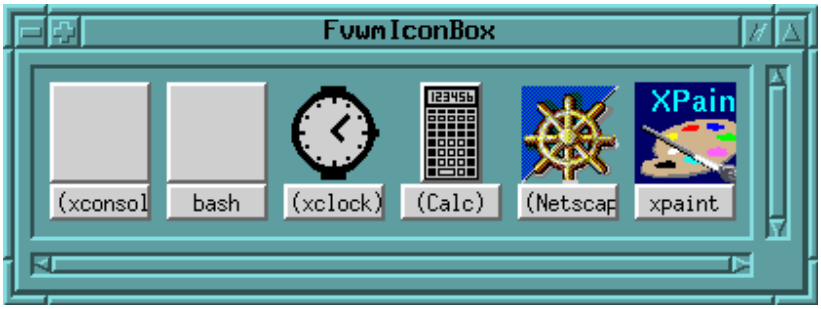

Figure 4-3: Example of an icon box

An icon manager or icon box simplifies icon management by keeping the icons together instead of having them spread out over the screen. Imagine the difficulty of finding a lonely icon hidden underneath a window.

Configuration File

The window manager's configuration file, sometimes referred to as its resource file or startup file, configures both the window manager's look and feel and its functionality. The configurable resources include the window manager's menus, key and pointer button functionality, colors, fonts, and much more.

The name of the configuration file differs from window manager to window manager. Each window manager has a systemwide configuration file, which can usually be found in a subdirectory of */etc/X11/* or */usr/X11R6/lib/X11/*. The filename usually begins with the word *system*, followed by a dot, then the name of the window manager, and then the letters *rc* (reminiscent of the Unix *rc* (run control) startup script). For example, *fvwm2's* configuration file is found in the directory */etc/X11/fvwm2/* (Red Hat) or */usr/X11R6/lib/X11/fvwm2/* (SuSE) and is named *system.fvwm2rc*.

The settings in the systemwide configuration file apply to all users of that window manager. In addition, you can create a personal configuration file for your window manager in your home directory that lets you override the systemwide settings and tailor the window manager to meet your needs.

Your personal configuration file has the same name as the systemwide file but without the *system* prefix. Hence your FVWM personal configuration file is named *.fvwm2rc*. If it is missing you can copy the systemwide file to your home directory, dropping the "*system*" prefix and modify it, or you can simply create a new one that contains only the settings you need. The settings in your personal file override those in the systemwide file. Anything not set in your file uses the system file setting.

Root Menu

Among the obvious visual distinctions between window managers is the *root menu* and how it is handled. A root menu is generally a *pop-up menu* that you see by pressing a pointer button in a blank area of the *root window*. The root menu usually has a default menu that can be changed by altering the window manager's configuration file (explained in the following section). Figure 4-4 is an example of a root menu.

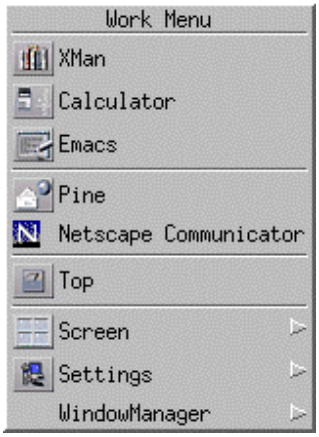

Figure 4-4: Example of a root menu

The root menu is usually selected using the right-most button of a three-button pointer. As is almost always the case in the X Window System, this is configurable in the window manager's configuration file, as described in the previous section.

Themes

A theme is a group of configuration settings and images that when used together provide an organized, aesthetic appearance. The combination of a pleasing choice of root window image and colors, coordinated with the style and color of the window frames and fonts results in an appealing theme.

Often a window manager provides pre-made, complete themes for you to choose from, or allows you to make your own themes. The same window manager can be made to look completely different using different themes. An interesting site for downloading themes is *http://www.themes.org*. Not only can you grab your choice of themes from this site, you're allowed to contribute your designed theme to the open source community.

Desktops and Workspaces

Consider a desktop being the root window, the displayed windows and icons on top of the root window, and possibly attached menus. If the window manager allows more than one desktop then it supports *virtual desktops*. This feature allows you to switch between separate desktops running different applications, Some window managers allow you to specify that an application should always appear, even if you switch desktops. For example, you might want a clock or calendar to show up regardless of which desktop you are working on. The clock application should run only once but appear on each desktop. Rather than being limited to the literal dimensions of your display screen, a window manager may support multiple screens or a single, larger-dimensioned screen. Such window managers are said to be *virtual window managers*. They allow the user to maintain and switch between multiple screens, each running different applications. Just like the capabilities of a virtual desktop, a single application can be set up to be visible on all the screens. Usually the number of screens, also known as pages, can be dynamically varied, and navigation among them is accomplished using a *panner*. The panner shows you the entire desktop in miniature, so you can keep track of what's going on in the large

desktop. An example of a panner, Motif's *pager*, is shown in Figure 4-5. From this figure it's apparent that application windows use only a small part of the available virtual screen space.

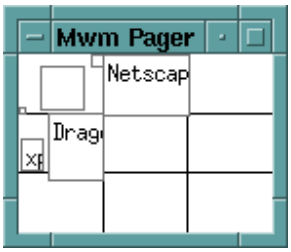

Figure 4-5: Motif's pager

One step even further is a virtual window manager supporting both multiple desktops and multiple screens per desktop.

Linux Window Managers

The following sections describe some of the more popular window managers available for Linux systems. The first three, Enlightenment, FVWM, and Window Maker, are described more fully in the next three chapters. Each description includes the URL of the window manager's homepage in parentheses at the end.

Enlightenment

Enlightenment, known to some simply as "E", is a window manager that was designed to be fast, extremely powerful, flexible, configurable, theme-enabled, and user-friendly. Enlightenment is often used as GNOME's window manager.

For more information concerning this window manager, see Chapter 7, *Enlightenment*. (*www.enlightenment.org*)

FVWM

FVWM, now in its second version as *fvwm2*, is a widely used window manager included in most Linux distributions. FVWM is a virtual window manager originally derived from *twm*. It is intended to have a small memory footprint and a rich feature set, be extremely customizable and extendable and have a high degree of Motif *mwm* compatibility. FVWM is highly extensible through its module interface. For more information concerning this window manager, see Chapter 5, *FVWM*. (*www.fvwm.org*)

Window Maker

Window Maker is a window manager designed to give integration support to the GNUstep* Desktop Environment. Even more so than AfterStep, Window Maker attempts in every way possible to reproduce the elegant look and feel of the NeXTSTEP GUI. It is fast, feature rich, easy to configure, and easy to use. In addition, Window Maker works with GNOME and KDE, making it one of the most universal window managers available.

For more information concerning this window manager, see Chapter 6, *Window Maker*. (*www.windowmaker.org*)

Other Window Managers

Here is a selection of other popular window managers. Each has strong supporters who appreciate its unique features. As you look through the list, you might find one that interests you or that has features you need. This is by no means a comprehensive list of Linux window managers. There are many more, and others are always being added (and some disappear). Hopefully this list will give you a sense of what's available and a starting point for finding a window manager that suits you.

3Dwm

3Dwm is a three-dimensional workspace manager[1] that can control Virtual Reality hardware such as CAVEs (Computer Aided Virtual Environments) and HMDs (Head-Mounted Device). It is a platform for research and development of three-dimensional user interfaces providing a means for exploring possible future user interfaces. It also provides backwards compatibility with legacy 2D X applications. (*www.3dwm.org*)

9wm

> 9wm is a simplistic window manager that attempts to emulate the Plan 9 operating system's 8-½ window manager as far as possible within the constraints imposed by X. It provides a simple yet comfortable user interface without garish decorations, title bars, or icons. (*dhog.g7.org/9wm.html*)

aewm

> aewm, the ascetic/aesthetic window manager, is a minimalist window manager. It has no nifty features, but is light on resources and extremely simple in appearance. It should eventually make a good reference implementation of the ICCCM. A few separate programs are included to handle running programs, switching between windows, etc. It does not provide any support for virtual desktops. (*www.red-bean.com/~decklin/aewm*)

---

* GNUstep is an object-oriented development environment that is meant to provide generalized visual interface design, a cohesive user interface, and look good. It provides an object-oriented application development framework and tool set for use on a wide variety of computer platforms. GNUstep is based on the original OpenStep specification published by NeXT Computer Inc., which now belongs to Apple Computer Inc.

[1] The 3Dwm homepage is careful to point out that the "'wm" in the name does not stand for "window manager".

AfterStep

AfterStep is a window manager for X that began as an emulation of NeXTSTEP*'s GUI but has been significantly altered according to the requests of various users. AfterStep aims to incorporate advantages of the NeXTSTEP interface and add additional useful features. Figure 4-6 should give you an impression of NeXTSTEP's graphical influence on AfterStep. (*www.afterstep.org*)

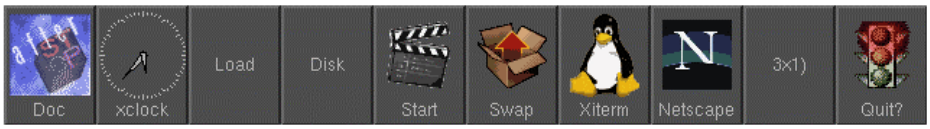

Figure 4-6: AfterStep

amiwm

amiwm (Amiga Window Manager) tries to make your display look and feel like an Amiga Workbench screen. It supports multiple screens, which can be dragged up and down just as on the Amiga, and has different backdrops for each session. (*www.lysator.liu.se/~marcus/amiwm.html*)

Blackbox

Blackbox is another window manager with a NeXTSTEP look and feel. It is designed to be small and fast, yet configurable. Blackbox's popularity is probably due to its dedication to efficiency. An example of its root menu is shown in Figure 4-7. (*blackbox.alug.org*)

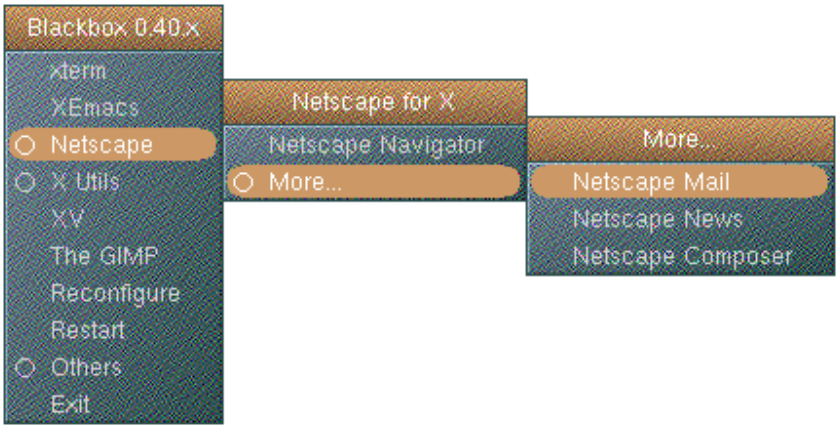

Figure 4-7: Blackbox's root menu

CTWM

CTWM (Claude's Tab Window Manager) is a highly configurable window manager based on the classic TWM window manager. It supports X pixmaps, up to 32 multiple virtual screens called workspaces, advanced icon management, animated

---

* NeXTSTEP's GUI is based on the OpenStep specification published by Steven Job's NeXT Computer Inc., which now belongs to Apple Computer Inc.

icons and backgrounds, 3D titles and borders. It offers rudimentary GNOME support and is backward compatible with TWM. (*ctwm.dl.nu*)

FLWM

FLWM (Fast Light Window Manager) is an X window manager designed to be user friendly and consume the minimum amount of screen real estate. It has no icons, instead using a pop-up root menu to select, hide or open new windows. FLWM is also designed to achieve GNOME, KDE, and Motif compatibility simultaneously. (*flwm.sourceforge.net*)

GWM

GWM, the Generic Window Manager, is an X11 window manager extensible via a built-in Lisp-like interpreter. (*koala.log.fr/gwm*)

IceWM

IceWM (The ICE Window Manager) is designed for speed, usability, and consistency. It is able to emulate the look of Motif, OS/2 Warp, Macintosh, or Microsoft Windows, and allows a customizable look using pixmaps. It is especially popular among users of Debian GNU/Linux. Pictured in Figure 4-8 is an *xclock* showing IceWM's window decoration using the "jim-mac" *theme*, which gives the window decoration a Macintosh look and feel. (*www.icewm.org*)

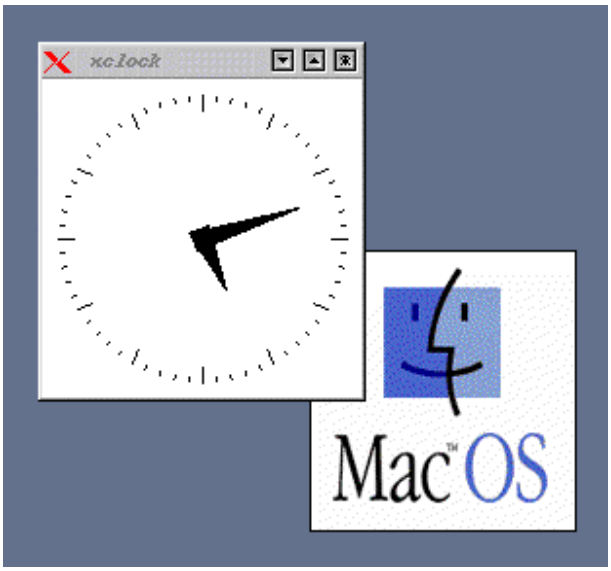

Figure 4-8: IceWM's "jim-mac" theme

MWM

mwm (Motif Window Manager) is a window manager that is compliant with the Motif desktop environment. It's a part of the commercial Open Group/Motif distribution. However, there is a free Motif clone appropriately called LessTif. (*www.opengroup.org/openmotif/*)

OLVWM

OLVWM (OpenLook Virtual Window Manager) is the virtual desktop version of OLWM, Sun Microsystems' OpenWindows Window Manager based on the Xview

toolkit. It allows the use of XPM pixmaps and GIF images as icons and menu images. (*ftp.x.org/R5contrib/olvwm.tar.Z*)

PerlWM
: PerlWM (Perl Window Manager) is a window manager written in Perl. It allows users to configure both the appearance and behavior of the window manager through Perl scripting. (*perlwm.sourceforge.net*)

PLWM
: PLWM, the Pointless Window Manager, is a highly modularized window manager written in Python. It has no configuration files; instead, you combine Python classes to make the perfect window manager for yourself. This is not a window manager for non-programmers. Currently the feature list includes point-to-focus and sloppy-focus; outline move, resizing, and deiconifying; views (extremely powerful workspaces); multihead support; and is fully keyboard-driven (with no mouse support at all). (*plwm.sourceforge.net*)

QVWM
: QVWM is a Microsoft Windows-like window manager. It allows Windows users to use X without a learning curve. (*www.qvwm.org*)

Sapphire
: Sapphire is small, fast, provides just enough features to get the job done, and tries to look nice in the process. (*sapphire.sourceforge.net*)

Sawfish
: Sawfish (formerly known as Sawmill) is an extensible window manager using a Lisp-based extension language. All window decorations are configurable and all user interface policy is controlled through the language. Its aim is to manage windows in the most flexible manner possible. It does not implement desktop backgrounds, application docks, or other things that may be achieved through separate applications. Sawfish is mostly GNOME-compliant and almost all configurations may be made though a graphical interface. It is often used as GNOME's window manager.

  For more information concerning this window manager see Chapter 8, *GNOME*. (*sawmill.sourceforge.net*)

SCWM
: SCWM, the Scheme Constraints Window Manager, is a highly dynamic and extensible window manager. SCWM embeds the *Guile* scheme as the configuration and extension language. Nearly all decorations can be changed at runtime on a per window basis. Dynamic loading of C modules is supported. SCWM is self-documenting and provides a powerful protocol for interacting with the window manager from other processes. (*scwm.sourceforge.net*)

SWM
: SWM, the Small Window Manager, was written for small computers with very little memory and small screen sizes. It is designed to run laptops (or even PDAs). SWM uses only Xlib and stdlib. (*www.small-window-manager.de*)

TWM
: *twm* (Tab Window Manager or Tom's Window Manager) was the default window manager on many X11 implementations in the past. It has been somewhat left behind by more recent window managers, making it something of historical interest. It was for some time the only real choice of window managers for Unix systems. Nearly

every window manager has since borrowed heavily from it and it is still distributed with XFree86. (There is no homepage, but archives are available at *www.xwinman.org/archive/twm*.)

VTWM
: VTWM, the Virtual Tab Window Manager, is a virtual window manager with adjustable graphical complexity. At its lower settings, it is ideal for limited-resource situations. It uses little memory, little CPU, few colors, and little bandwidth. (*www.visi.com/~hawkeyd/vtwm.html*)

wm2
: wm2 (A Window Manager) is a simple window manager that supports a single desktop. It adds a stylish frame to each managed window as shown in Figure 4-9. The follow-on to wm2 is wmx, which supports multiple desktops. (*www.all-day-breakfast.com/wm2/*)

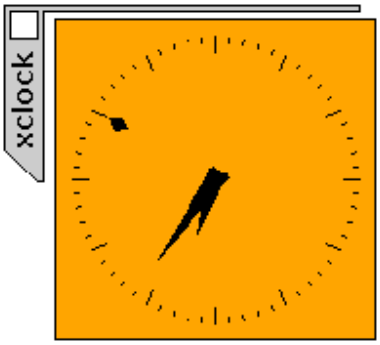

Figure 4-9: wm2's stylish frame

YAWM
: YAWM, the Yet Another Window Manager, is a lightweight yet very productive window manager. Another important goal is portability across UNIX platforms through use of standard libraries (POSIX and Xlib). (*yawm.sourceforge.net*)

Desktop Environments

Desktop environments are a leap beyond window managers. A desktop environment is a completely integrated GUI desktop. It provides integrated applications and a common programming environment. Desktop environments such as GNOME and KDE are object-based, and they provide object request brokers (*ORB*s) to manage the objects. This makes possible the high level of interoperability that the desktop environments are noted for. Some of the desktop environment's main features include:

Consistent look and feel
: All applications use similar menu layouts, icons, scrollbars, etc. Instead of allowing multiple application windows that might each have its own menu layout, key bindings, and appearance, the desktop environment imposes consistency. Not only does this give the windows a consistent look, but it greatly simplifies learning to use new applications.

Common drag-and-drop and cut-and-paste protocols
: Applications may interchange objects among themselves. For example, you can drag a file to a folder to move the file to that folder, or drag a file to an application icon to start that application using the dragged file. Similarly, you can cut or copy objects from one application and paste them into another application. (A window manager lets you cut-and-paste text between windows, but it doesn't let you reuse objects, such as figures.)

Dialog-based desktop configuration
: Mouse-driven configuration applications are provided instead of requiring you to edit configuration files. Each desktop environment provides a centralized application to do all kinds of configuration, so you don't need to edit multiple files or run several different configuration applications.

Common application-development framework
: A desktop environment provides a programming environment in the form of libraries and tools for building desktop-compliant applications. Furthermore, the same libraries are shared among the applications, so loading in a new application doesn't also require loading in new libraries..

Compound documents
: Documents can contain objects that are shared by several applications.

Common session management
: Desktop environments provide a uniform mechanism for saving and restoring sessions. When you log out, if you indicate that you want your session saved (usually by checking a box in the logout dialog), the session manager stores the internal state of each running application.

Common file management and help systems
: Desktop environments provide file management applications for managing files and directories. They also provide help systems that display not only their own documentation, but also man pages and info pages. Both file management and help are generally browser-based, so you can work across the network as easily as on your own system.

Panels, application launchers, and applets
: Many of the notable features of the desktop environments have to do with their architecture, and are not directly visible to users except as they are reflected in behaviors such as drag-and-drop. The features listed here are some of the more visible aspects of the desktop environment. The panel provides a place to put menus, the pager, and application launchers. An application launcher is a button that starts an application, and an applet is a small desktop-compliant application.

Linux Desktop Environments

The two most popular desktop environments on Linux are GNOME and KDE. Most of the major distributions install one or the other of these environments by default. Both are comprehensive, full-featured environments. While you can run some applications from one under the other, they are based on different toolkits and thus are not truly interoperable. You can't, for example, drag a Gnome object to a KDE application and expect it to run. The best thing to do is to try them both and see which you prefer.

GNOME

GNOME (pronounced “Gah-NOME”, the GNU Network Object Model Environment) is a freely available desktop that can be used with any of several window managers, including Enlightenment and Sawfish. GNOME is based on the GTK+ toolkit and is open source software.

For much more information concerning this desktop environment see Chapter 8, *GNOME*. (*www.gnome.org*)

KDE

KDE is a freely available desktop that includes the K Window Manager and a large variety of accessories and applications that is constantly growing. KDE is freely redistributable and uses the QT Widget set which until recently was distributed under a non-free license. KDE 2.0 has now been released and recent versions of QT are available without royalty.

For much more information concerning this desktop environment see Chapter 9, *KDE*. (*www.kde.org*)

Other Desktop Environments

GNOME and KDE are not the only desktop environments used in today’s Linux world. An additional two are CDE and XFce.

CDE

CDE (Common Desktop Environment) is the desktop that was intended to unify the GUI for most implementations of commercial Unix. It provides a virtual desktop and is also available for Microsoft NT. Though available for Linux, CDE is not free. It incorporates the *dtwm* window manager, which is a Motif-compliant, virtual window manager. (*news://comp.unix.cde*)

XFce

XFce is a lightweight desktop environment for Linux systems. It’s similar to the commercial CDE, and is based on the GTK+ toolkit.

This desktop environment includes a window manager, called XFwm, a control panel, a file manager, a backdrop manager, a sound manager, a calendar, a pager module, and a GNOME compliance module. (*www.xfce.org*)

Summary

We’ve looked at both window managers and desktop environments. How do they fit together? A window manager stands by itself and doesn’t require the use of a desktop environment, as long as you can live without the additional features of a desktop environment. A desktop, however, requires a window manager. It doesn’t replace the window manager functionality, but builds upon it.

KDE provides its own window manager, KWM, by default, while GNOME defaults to either Enlightenment or Sawfish, as we've seen. With either KDE or GNOME, you can modify the default to run a window manager that you prefer; depending on how compliant that window manager is, you'll find more or fewer of the desktop features available.

In fact, the distinction between the two is getting fuzzy, as some of the more advanced window managers start to incorporate some of the features of the desktops, such as session management, panels, application launching, and the docking (or "swallowing") of applications into the panel.

The desktops offer an advanced working environment and for many—perhaps most—users, they are the way to go. But they do have some disadvantages as well. The major one is size and complexity. Because of their many features, the desktop environments are necessarily large and resource-intensive. This can impact performance, particularly on an older or smaller system. If you are running X on such a system, you might want to take a look at Xfce, or one of the fancier window managers.

Also, both KDE and GNOME have complex menu structures that can make it difficult to find an application until you've learned your way around. And they both duplicate many of the programs that are already available in X—this is necessary for compliance, but does lead to some redundancy.

Finally, as we mentioned earlier, GNOME and KDE use different toolkits and different ORBs and therefore aren't interoperable. There have been efforts to get them to work together, but the outcome of any such efforts is uncertain. As long as one or the other meets your needs, this shouldn't be a problem for you.

The remaining chapters of Part II describe a few of the major window managers (FVWM, Window Maker, and Enlightenment) and the desktop environments GNOME and KDE. Clearly these are only a few of the many possibilities, but understanding what they offer and how to configure and work with these programs should give you a foundation for selecting and working with whichever alternative window manager or desktop you prefer.