# LPI exam 101 prep: The X Window System

## Junior Level Administration (LPIC-1) topic 110

Skill Level: Intermediate

Ian Shields (ishields@us.ibm.com)
Senior Programmer
IBM

02 Jul 2006

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 101. In this fifth in a series of five tutorials, Ian introduces you to the X Window System on Linux®. By the end of this tutorial, you will know how to install and maintain the X Window System. This tutorial covers both major packages for X on Linux: XFree86 and X.Org.

# Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

## About this series

The Linux Professional Institute (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 101, the five topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 101: Tutorials and topics

| LPI exam 101 topic | developerWorks tutorial | Tutorial summary |
|---|---|---|
| Topic 101 | LPI exam 101 prep: Hardware and architecture | Learn to configure your system hardware with Linux. By the end of this tutorial, you will know how Linux configures the hardware found on a modern PC and where to look if you have problems. |
| Topic 102 | LPI exam 101 prep: Linux installation and package management | Get an introduction to Linux installation and package management. By the end of this tutorial, you will know how Linux uses disk partitions, how Linux boots, and how to install and manage software packages. |
| Topic 103 | LPI exam 101 prep: GNU and UNIX commands | Get an introduction to common GNU and UNIX commands. By the end of this tutorial, you will know how to use commands in the bash shell, including how to use text processing commands and filters, how to search files and directories, and how to manage processes. |
| Topic 104 | LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard. | Learn how to create filesystems on disk partitions, as well as how to make them accessible to users, manage file ownership and user quotas, and repair filesystems as needed. Also learn about hard and symbolic links, and how to locate files in your filesystem and where files should be placed. |
| Topic 110 | LPI exam 101 prep: The X Window system | (This tutorial). Learn how to install and maintain the X Window System. See detailed objectives below. |

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger

system, back up and restore, and shut down and reboot

- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the developerWorks tutorials for LPI exam 101, as well as the entire set of developerWorks LPI tutorials.

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

## About this tutorial

Welcome to "The X Window System," the fifth of five tutorials designed to prepare you for LPI exam 101. In this tutorial, you learn about setting up the X Window System on Linux. This tutorial covers both major packages for X on Linux: XFree86 and X.Org.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. The X Window System: Exam objectives covered in this tutorial

| LPI exam objective | Objective weight | Objective summary |
|---|---|---|
| 1.110.1 Install and configure X | Weight 5 | Configure and install X and an X font server. Check that the video card and monitor are supported by your X server, and customize and tune X for the card and monitor. Install an X font server, install fonts, and configure X to use the font server. |
| 1.110.2 Set up a display manager | Weight 3 | Set up and customize a display manager. Turn the display manager on or off and change its greeting and default bitplanes. Configure display managers for use by X stations, such as the X, GNOME, and KDE display managers. |
| 1.110.4 Install and customize a window manager environment | Weight 5 | Customize a system-wide desktop environment and window manager, including window manager menus and desktop panel menus. Select and configure an X terminal, and verify and resolve library |

| | |
|---|---|
| | dependency issues for X applications. Export an X display to a client workstation. |

## Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial. You should also be familiar with using GUI applications, preferably under the X Window System.

This tutorial builds on content covered in the previous four tutorials in this series, so you may want to first review the tutorials for topics 101, 102, 103, and 104.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

---

# Section 2. Install and configure X

This section covers material for topic 1.110.1 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 5.

In this section, you learn how to:

- Check that your video card and monitor are supported by your X server
- Configure and install X
- Customize and tune X for the card and monitor
- Configure and install an X font server
- Install fonts

## History of the X Window System

The X Window System, also known simply as X or X11, is a window system for graphical (bitmap) displays. X originated at MIT in 1984 and was developed as part of Project Athena, which provided a computing environment using disparate hardware. X separates the display functions into a *display server* and *clients*, which provide the application logic. It is network transparent, so the display server and the

client need not be on the same machine. Note that the sense of "client" and "server" is somewhat opposite to what you might normally think. In addition to handling displayed output, the server end also handles input from devices such as keyboards, mice, graphic tablets, and touchscreens.

X provides a toolkit for GUI applications, but it does not specify a user interface. On a typical Linux system, you will choose between KDE or GNOME desktops, and you may have several other window managers available too. Because X does not specify a user interface, these desktops and window managers have different appearances.

Because X was developed to serve a large community with disparate hardware types, you will find that different versions of X client and server will generally interoperate quite well.

### XFree86 and X.Org

By 1987, MIT wanted to hand off control of X, and the MIT X Consortium was founded as a non-profit group to oversee development of X. After a few more stewardship changes, the Open Group formed X.Org in 1999. Since 1992, much of the active development of X was done by XFree86, which had originally created a port of X to the Intel® 386 hardware for use in Linux, hence the name XFree86. XFree86 joined X.Org as a non-paying member.

Although originally created for the 386, later versions of XFree86 supported several different platforms, and it became the most widely used X version on Linux. After some disputes over new licensing terms and the development model of XFree86, the X.Org Foundation was formed. Working from the last XFree86 version under the earlier license, it created X11R6.7 and X11R6.8. Many distributions still use XFree86, while many have picked up X.Org instead.

## Video hardware support

Both the XFree86 and X.Org packages support a wide range of modern video cards. Consult the online documentation for your release (see Resources). Some manufacturers do not release open source drivers for all functions, so you may need to integrate a driver from the manufacturer into your XFree86 system. Check the manufacturer's Web site for improved or updated Linux drivers. This is often the case for accelerated 3D drivers. Even if the hardware capabilities of your card cannot be used by XFree86, it is possible that you may be able to run in VESA (Video Electronics Standards Association) *framebuffer* mode.

Modern monitors implement the VESA *Display Data Channel* (or *DDC*) specification, which allows monitor information and capabilities to be determined programatically. The XFree86 configuration tools (other than `xf86config`) use this information to

configure your X system.

One way to see how X works with your hardware is to boot a live CD distribution, such as Knoppix or Ubuntu. These generally have excellent ability to detect and use your hardware. Many distributions offer a graphical installation choice, which also requires correct detection and use of your hardware.

## XFree86

Most distributions include a version of XFree86 or X.Org already packaged for the system. If not, you may be able to find an RPM or .deb package and install it according to the techniques you learned in the tutorial for topic 102, "LPI exam 101 prep: Linux installation and package management."

### XFree86 installation

If you do not have an XFree86 package available, then you will need to download the files from XFree86 project Web site (see Resources). Prebuilt packages are available for Linux on several popular hardware platforms, or you may install from the source distribution. This tutorial assumes you will be installing a binary package of the current release (version 4.5.0).

You will need to download several binary packages. You should use the available md5 checksums and the GPG keys to validate your downloads. Table 3 lists the required files for XFree86.

Table 3. XFree86 required files

| File | Description |
|---|---|
| Xinstall.sh | Installation script |
| extract | Tarball extraction utility |
| Xbin.tgz | X clients, utilities, and run-time libraries |
| Xlib.tgz | Data files required at run-time |
| Xman.tgz | Manual pages |
| Xdoc.tgz | XFree86 documentation |
| Xfnts.tgz | Base set of fonts |
| Xfenc.tgz | Font encoding data |
| Xetc.tgz | Run-time configuration files - part 1 |
| Xrc.tgz | Run-time configuration files - part 2 |

| | |
|---|---|
| Xvar.tgz | Run-time data |
| Xxserv.tgz | XFree86 X server |
| Xmod.tgz | X server modules |

If you are not sure which version to download, then download the Xinstall.sh file for the one you think is closest, and use the `-check` option to check your system as shown in Listing 1.

**Listing 1. Checking for the correct XFree86 binary package**

```
root@pinguino:~/xfree86# sh Xinstall.sh -check
Checking which OS you're running...
uname reports 'Linux' version '2.6.12-10-386', architecture 'i686'.
libc version is '6.3.5' (6.3).

Binary distribution name is 'Linux-ix86-glibc23'

If you don't find a binary distribution with this name, then
binaries for your platform are not available from XFree86.org.
```

For this example, you should look for the "Linux-ix86-glibc23" package.

Table 4 lists the optional files for XFree86. For this tutorial, you will need the font server and any other items that you wish to install.

Table 4. XFree86 optional files

| File | Description |
|---|---|
| Xdrm.tgz | Direct rendering manager (DRM) kernel modules source |
| Xfsrv.tgz | Font server |
| Xnest.tgz | Nested X server |
| Xprog.tgz | X header files, configuration files, and libraries for developing X applications |
| Xprt.tgz | X Print server |
| Xvfb.tgz | Virtual framebuffer X server |
| Xtinyx.tgz | TinyX servers |
| Xf100.tgz | 100dpi fonts |
| Xfcyr.tgz | Cyrillic fonts |
| Xfscl.tgz | Scalable fonts (Speedo, Type1, and TrueType) |

| | |
|---|---|
| Xhtml.tgz | HTML version of the documentation |
| Xps.tgz | PostScript version of the documentation |
| Xpdf.tgz | PDF version of the documentation |

Before you install XFree86, you should make backups of your /usr/X11R6, /etc/X11, and /etc/fonts directories as their contents may be changed by the XFree86 installation. You may use the `tar`, `cp`, or `zip` commands to do this. When you are ready to install XFree86, change to the directory where you downloaded the XFree86 files, and run the Xinstall.sh script as shown in Listing 2.

**Listing 2. Installing XFree86**

```
root@pinguino:~/xfree86# sh Xinstall.sh
```

You will be prompted for answers to several questions, which may vary according to whether or not you have a prior installation of X. After the mandatory components are installed, you will be prompted to install the optional components individually.

Following the file installation, the script will run the `ldconfig` command and offer to set up several symbolic links for you.

The easiest way to install XFree86 is to install all the components you want using the Xinstall.sh script. If you do not, you will either need to reinstall the whole package, potentially overwriting any customization you have done, or manually install other components.

### XFree86 configuration

Historically, configuring XFree86 involved creating an XF86Config file, which contained information about the video card, mouse, keyboard, and display hardware as well as customization items, such as preferred display resolutions. The original configuration tool, `xf86config`, required a user to have and enter detailed information about video card and monitor timings. Recent versions of XFree86 are capable of dynamically determining the available hardware and can run with little or no configuration information.

The available configuration tools are:

**XFree86 -autoconfig**

Running `XFree86` with the `-autoconfig` option will attempt to automatically configure the X server. If your setup is correctly determined, you should be able to move the X cursor around the screen with your mouse. Hold down the **Ctrl**

and **Alt** keys and press the **Backspace** key to exit display. This confirms that automatic configuration will work. A configuration file is not written.

**XFree86 -configure**

Running `XFree86` with the `-configure` option may work if the `-autoconfig` option does not. This option may also give problems on some systems.

**xf86cfg**

The `xf86cfg` command attempts to start the display and input drivers. If it is successful, you will see a window with a diagram of your system. Right-click an item to view or update its configuration. On some systems you may need to use the numeric keypad instead of mouse buttons, because the mouse was not properly detected. You may wish to try creating a symbolic link to /dev/mouse from your actual mouse device before running `xf86cfg`. For example:

```
ln -s /dev/input/mice /dev/mouse
```

When you click **Quit**, you will be prompted to save your /etc/X11R6/lib/X11/XF86Config and /etc/X11R6/lib/X11/xkb/X0-config.keyboard configuration files.

**xf86config**

The `xf86config` command uses a text-mode interface to interactively prompt for information about your mouse, keyboard, video card, and display. You will need horizontal and vertical frequency information for your display. You can select most video cards from a database of known video cards. If not, you may need specific chipset and timing information for your card.

**Notes:**

1. If your system includes XFree86, your distributor may have included a tool, such as the `sax2` command used on SUSE systems or the `redhat-config-xfree86` command used on some Red Hat® systems. Always check your system documentation for such tools.

2. Another configuration tool, `XF86Setup`, is no longer distributed with XFree86.

## X.Org

Most distributions include a version of XFree86 or X.Org already packaged for the system. If not, you may be able to find an RPM or .deb package and install it according to the techniques you learned in the tutorial for topic 102, "LPI exam 101 prep: Linux installation and package management."

**X.Org installation**

If you do not have an X.Org package available, then you will need to download and build the source from the X.Org Web site or a mirror (see Resources). At the time of this writing, these sites do not contain prebuilt binary packages for X11R6.9.0 or X11R7.0. The source is available from the CVS repository, or as tarballs that are compressed with either gzip or bzip2. You need to get either the gz or bz2 files, **not both**. You will find the *X.Org Modular Tree Developer's Guide* (see Resources) an invaluable aid when downloading and building X.Org yourself. Take note of the additional packages, such as freetype, fontconfig, and Mesa, that are recommended for a fully functional build.

### X.Org configuration

The X.Org package is based on a recent version of XFree86 and has similar configuration capabilities, including dynamically determining the available hardware. The configuration file is named xorg.conf rather than XF86Config. You may find it in one of several places: /etc/xorg.conf, /etc/X11/xorg.conf, /usr/X11R6/etc/xorg.conf, /usr/X11R6/lib/X11/xorg.conf.hostname, or /usr/X11R6/lib/X11/xorg.conf.

The available configuration tools are:

**X -configure**
: Running X with the `-configure` option causes the X server to load each driver module, probe for the driver, and create a configuration file that is saved in the home directory of the user who started the server (usually /root). The file is called xorg.conf.new.

**xorgcfg**
: This tool is similar to xf86cfg

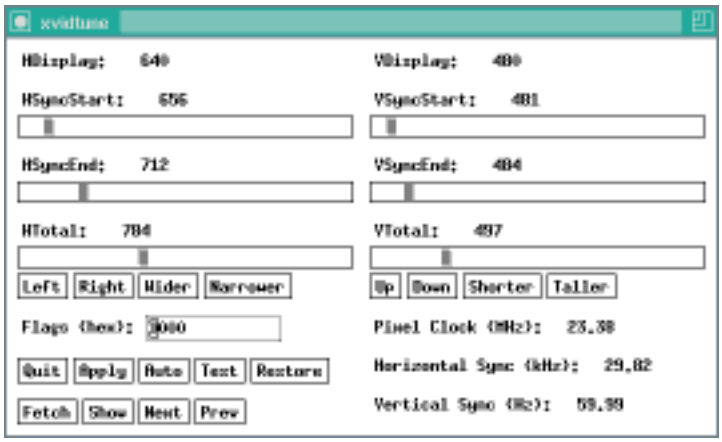
**xorg86config**
: The `xorgconfig` command uses a text-mode interface to interactively prompt for information about your mouse, keyboard, video card, and display. As with xf86config, you will need horizontal and vertical frequency information for your display. You can select most video cards from a database of known video cards. If not, you may need specific chipset and timing information for your card.

## Tuning X

Modern multisync CRT monitors usually have controls to set the size and position of the displayed image on the screen. If your monitor does not have this capability, you can use the `xvidtune` command to tune the size and position of your X display. When you run `xvidtune` from an X terminal session, you will see a window similar to Figure 1. Adjust the settings and click **Test** to see how they work, or click **Apply** to change the settings. If you click **Show**, the current settings will be printed to your

terminal window in a format that you can use as a Modeline setting in your CF86Config or xorg.conf file.

**Figure 1. Running xvidtune**




Consult the man page for additional information.

## Overview of fonts in X

For many years, font handling on X systems was done by the *core X11 fonts system*. Recent versions of XFree86 (and X.Org) X servers include the *Xft fonts system*. The core fonts system was originally designed to support monochrome bitmap fonts, but it has been enhanced over time. The Xft system was designed to handle modern requirements, including anti-aliasing and sub-pixel rasterization and it allows applications to have extensive control over the rendering of glyphs. A major difference between the two systems is that the core fonts are handled on the server, while the Xft fonts are handled by the clients, which send the necessary glyphs to the server.

X originally used Type 1 (or Adobe Type 1) fonts, a font specification developed by Adobe. The Xft system can handle these as well as OpenType, TrueType, Speedo, and CID font types.

### The xfs font server

With the core X11 fonts system, the X Server obtains fonts and font information from a *font server*. The X font server, `xfs`, usually runs as a daemon and is started at system startup, although it is possible to run it as an ordinary task. You will usually install a font server as part of your X installation. However, because X is a network protocol, it is possible to obtain fonts and font information over a network rather than from your local machine.

The X font server uses a configuration file, normally /usr/X11R6/lib/X11/fs/config. A

sample font configuration file is shown in Listing 3. The configuration file may also be located in or linked to /etc/X11/fs.

**Listing 3. Sample /usr/X11R6/lib/X11/fs/config**

```
# allow a max of 10 clients to connect to this font server
client-limit = 10

# when a font server reaches its limit, start up a new one
clone-self = on

# alternate font servers for clients to use
#alternate-servers = foo:7101,bar:7102

# where to look for fonts
#
catalogue = /usr/X11R6/lib/X11/fonts/misc:unscaled,
        /usr/X11R6/lib/X11/fonts/75dpi:unscaled,
        /usr/X11R6/lib/X11/fonts/100dpi:unscaled,
        /usr/X11R6/lib/X11/fonts/misc,
        /usr/X11R6/lib/X11/fonts/Type1,
        /usr/X11R6/lib/X11/fonts/Speedo,
        /usr/X11R6/lib/X11/fonts/cyrillic,
        /usr/X11R6/lib/X11/fonts/TTF,
        /usr/share/fonts/default/Type1

# in 12 points, decipoints
default-point-size = 120

# 100 x 100 and 75 x 75
default-resolutions = 75,75,100,100

# how to log errors
use-syslog = on

# don't listen to TCP ports by default for security reasons
no-listen = tcp
```

This example is typical of a Linux workstation installation where the font server does not provide fonts over TCP network connections (`no-listen = tcp`).

### The Xft library

The Xft library provides functions that allow client applications to select fonts based on pattern criteria and to generate glyphs to send to the server. Patterns take into account items such as font family (Helvetica, Times, and so on), point size, weight (regular, bold, italic), and many other possible characteristics. While the core font system allowed a client to find a match for the first available font on a server, the Xft system finds the best match for all the criteria and then sends the glyph information to the server. Xft interacts with FreeType for font rendering and with the X Render extensions, which help speed up font rendering operations. Xft is included with current versions of both XFree86 and X.Org.

**Note:** If your X server is running across a network and using a video card that does not support the X Render extensions, you may want to disable anti-aliasing as the network performance in this case may be a problem. You can use the `xdpyinfo`

command to check your X server. Listing 4 shows part of the output from `xdpyinfo`. Since the output from `xdpyinfo` is large, you may want to use `grep` to filter for 'RENDER'.

**Listing 4. Checking for RENDER extensions with xdpyinfo**

```
[ian@lyrebird ian]$ xdpyinfo
name of display:    :0.0
version number:    11.0
vendor string:    The XFree86 Project, Inc
vendor release number:    40300000
XFree86 version: 4.3.0
maximum request size:  4194300 bytes
motion buffer size:  256
bitmap unit, bit order, padding:    32, LSBFirst, 32
image byte order:    LSBFirst
number of supported pixmap formats:    7
supported pixmap formats:
    depth 1, bits_per_pixel 1, scanline_pad 32
    depth 4, bits_per_pixel 8, scanline_pad 32
    depth 8, bits_per_pixel 8, scanline_pad 32
    depth 15, bits_per_pixel 16, scanline_pad 32
    depth 16, bits_per_pixel 16, scanline_pad 32
    depth 24, bits_per_pixel 32, scanline_pad 32
    depth 32, bits_per_pixel 32, scanline_pad 32
keycode range:    minimum 8, maximum 255
focus:  window 0x2000011, revert to Parent
number of extensions:    30
    BIG-REQUESTS
    DOUBLE-BUFFER
    DPMS
    Extended-Visual-Information
    FontCache
    GLX
    LBX
    MIT-SCREEN-SAVER
    MIT-SHM
    MIT-SUNDRY-NONSTANDARD
    RANDR
    RECORD
    RENDER
    SECURITY
    SGI-GLX
    SHAPE
    SYNC
    TOG-CUP
    X-Resource
    XC-APPGROUP
    XC-MISC
    XFree86-Bigfont
    XFree86-DGA
    XFree86-DRI
    XFree86-Misc
    XFree86-VidModeExtension
    XInputExtension
    XKEYBOARD
    XTEST
    XVideo
default screen number:    0
number of screens:    1
```

The use of Xft instead of the core X system requires application changes, so you may find that some of your applications do not appear to take advantage of the

better font rendering of Xft. At the time of this writing, the Qt toolkit (used for KDE) and the GTK+ toolkit (used for GNOME), as well as Mozilla 1.2 and above, are examples of applications that use Xft.

## Installing fonts

There are two methods of installing fonts, one for Xft and a more complex one for the core X11 fonts.

### Fonts for Xft

Xft uses fonts located in a set of well-known font directories as well as fonts installed in the .fonts subdirectory of the user's home directory. The well-known font directories include subdirectories of /usr/X11R6/lib/X11/lib/fonts as listed in the catalog entry in /usr/X11R6/lib/X11/fs/config. Other font directories may be specified in the FontPath section XF86Config or xorg.conf, according to which X Server package you are using.

Simply copy your font files to the user's .fonts directory or to a directory such as /usr/local/share/fonts for system-wide use. The font server should pick up the new fonts and make them available at its next opportunity. You can trigger this update with the `fc-cache` command.

The current font technology in X uses loadable modules to provide font support for different font types as shown in Table 5.

Table 5. X Server font modules

| Module | Description |
|---|---|
| bitmap | bitmap fonts (.bdf, .pcf, and .snf) |
| freetype | TrueType fonts (.ttf and .ttc), OpenType fonts (.otf and .otc) and Type 1 fonts (.pfa and .pfb) |
| type1 | Alternate for Type 1 (.pfa and .pfb) and CID Fonts |
| xtt | Alternate TrueType module (.ttf and .ttc) |
| speedo | Bitstream Speedo fonts (.spd) |

If you are having trouble installing and using a font, check the server log (for

example, /var/log/XFree86.0.log) to make sure that the appropriate module was loaded. Module names are case sensitive. You can use the `xset` command to display (and set) X server settings, including the font path and the location of the configuration and log files, as illustrated in Listing 5.

**Listing 5. Displaying X server settings with xset**

```
[ian@lyrebird ian]$ xset -display 0:0 -q
Keyboard Control:
  auto repeat:  on    key click percent:  0    LED mask:  00000000
  auto repeat delay:  500    repeat rate:  30
  auto repeating keys:  00ffffffdffffbbf
                        fadfffffffdfe5ff
                        fffffffffffffffe
                        ffffffffffffffff
  bell percent:  50    bell pitch:  400    bell duration:  100
Pointer Control:
  acceleration:  2/1    threshold:  4
Screen Saver:
  prefer blanking:  yes    allow exposures:  yes
  timeout:  0    cycle:  0
Colors:
  default colormap:  0x20    BlackPixel:  0    WhitePixel:  16777215
Font Path:
  /home/ian/.gnome2/share/cursor-fonts,unix/:7100,/home/ian/.gnome2/share/fonts
Bug Mode: compatibility mode is disabled
DPMS (Energy Star):
  Standby: 7200    Suspend: 7200    Off: 14340
  DPMS is Enabled
  Monitor is Off
Font cache:
  hi-mark (KB): 5120  low-mark (KB): 3840  balance (%): 70
File paths:
  Config file:  /etc/X11/XF86Config
  Modules path: /usr/X11R6/lib/modules
  Log file:     /var/log/XFree86.0.log
```

If you need to further control the behavior of Xft, you can use either the system-wide configuration file (/etc/fonts/fonts.conf) or a user-specific file (.fonts.conf in the home directory of the user). You can enable or disable anti-aliasing and control sub-pixel rendering (used on LCD displays), among other things. These are XML files, so you need to ensure that you maintain well-formed XML if you edit them. Consult the man (or info) pages for fonts-conf for additional information on the contents and format of these files.

### Core X11 fonts

If you have files in Bitmap Distribution Format (.bdf), it is desirable to convert them to Portable Compiled Format (.pcf) using the `bdftopcf` command and then compress them using `gzip` before installing them. Once this is done, you may copy your new fonts to a directory, such as /usr/local/share/fonts/bitmap/, and then run the `mkfontdir` command to create a font directory for use by the font server. These steps are illustrated in Listing 6.

**Listing 6. Installing a bitmapped font**

```
[root@lyrebird root]# bdftopcf courier12.bdf -o courier12.pcf
[root@lyrebird root]# gzip courier12.pcf
[root@lyrebird root]# mkdir -p /usr/local/share/fonts/bitmap
[root@lyrebird root]# cp *.pcf.gz /usr/local/share/fonts/bitmap/
[root@lyrebird root]# mkfontdir /usr/local/share/fonts/bitmap/
[root@lyrebird root]# ls /usr/local/share/fonts/bitmap/
courier12.pcf.gz  fonts.dir
```

Note that the `mkfontdir` command created the fonts.dir file.

When installing scalable fonts, such as TrueType or Type1 fonts, an extra step is required to create scaling information. After you copy the font files to the target directory, run the `mkfontscale` command and then the `mkfontdir` command. The `mkfontscale` command will create an index of scalable font files in a file called fonts.scale.

Once you have set up the font directory and scaling information for your new fonts, you need to tell the server where to find them, by including the new directory in the font path. You can do this temporarily by using the `xset` command or permanently by adding a FontPath entry to XF86Config or xorg.conf. To add the new bitmap font directory to the front of the font path, use the `+fp` option of `xset` as shown in Listing 7.

**Listing 7. Updating the fontpath with xset**

```
[ian@lyrebird ian]$  xset +fp /usr/local/share/fonts/bitmap/ -display 0:0
```

Although not shown here, it is a good idea to add scalable fonts before bitmapped fonts in the path as this results in better font matching. To add directories to the back of the path, use the `fp+` option. Similarly, the use of `-fp` and `fp-` options will remove font directories from the front and back of the font path, respectively.

You can make permanent modifications to the font path by editing the XF86Config or xorg.conf file. You can add as many FontPath lines as you need in the Files section, as shown in Listing 8.

**Listing 8. Updating XF86Config or xorg.conf**

```
Section "Files"
# RgbPath is the location of the RGB database.  Note, this is the name of the
# file minus the extension (like ".txt" or ".db").  There is normally
# no need to change the default.

# Multiple FontPath entries are allowed (they are concatenated together)
# By default, Red Hat 6.0 and later now use a font server independent of
# the X server to render fonts.

        RgbPath      "/usr/X11R6/lib/X11/rgb"
        FontPath     "unix/:7100"
```

```
        FontPath     "/usr/local/share/fonts/bitmap/"
EndSection
[
```

For information on other modifications you can make to the X configuration files, see the man pages for either XF86Config or xorg.conf, as appropriate.

---

# Section 3. Set up a display manager

This section covers material for topic 1.110.2 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 3.

In this section, you learn how to:

- Set up and customize a display manager
- Change the display manager greeting
- Change default bitplanes for the display manager
- Configure display managers for use by X stations

The display managers covered are XDM (X Display Manager), GDM (GNOME Display Manager), and KDM (KDE Display Manager).

## Display managers

In the previous section, if you installed and configured X on a system that did not have X already installed, you probably noticed that, to get any kind of graphical display, you must initially log in to a terminal window and run the `startx` command. While this works for the local display, it is cumbersome. Furthermore, it does not work for a remote X terminal.

The solution is to use a *display manager* to present a graphical login screen and handle authentication. Once a user authenticates, the display manager starts a session for the user on the system where the display manager is running. The graphical output is displayed on the screen where the user entered his or her login credentials. This may be a local display or an X display connected across a network.

Both XFree86 and X.Org come with the XDM display manager. Two other display managers are also popular, KDE and GNOME. In this section, you learn how to set up and customize these three display managers.

To set up a graphical login, however, you need to understand Linux system initialization. You can learn more about this in the forthcoming tutorial, LPI exam 102 prep (topic 106): Boot, initialization, shutdown, and runlevels, and in LPI exam 201 prep (topic 202): System startup. In the remainder of this section, you will learn enough to start your system with a graphical login, but the main focus in this section is on setting up and customizing the display manager.

On systems such as Red Hat® and SUSE systems, X is usually started in runlevel 5. Debian systems treat runlevels 2 through 5 as equivalent and default to starting in runlevel 2. The determination of default runlevel is made in /etc/inittab as shown in Listing 9.

**Listing 9. Setting the default runlevel in /etc/inittab.**

```
# The default runlevel is defined here
id:5:initdefault:
```

Another line, such as that shown in Listing 10 (for a SUSE system) or Listing 11 (for a Ubuntu system), determines the script or program to be run first.

**Listing 10. Initial script for SUSE (or Red Hat) system**

```
# First script to be executed, if not booting in emergency (-b) mode
si::bootwait:/etc/init.d/boot
```

**Listing 11. Initial script for Ubuntu (or Debian) system**

```
# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS
```

The initialization scripts (/etc/init.d/boot or /etc/init.d/rcS) will then run other scripts. Eventually, a series of scripts for the chosen runlevel will be run. For the above examples, these might include /etc/rc2.d/S13gdm (Ubuntu) or /etc/init.d/rc5.d/S16xdm (SUSE), which are both scripts to run a display manager. You will find that the rc*n*.d directories in /etc/init.d usually contain symbolic links to scripts in /etc/init.d without the leading S (or K) and number. The **S** indicates that the script should be run when the runlevel is entered, and the **K** indicates that the script should be run when the runlevel is terminated. The digits specify an order from 1 to 99 in which the scripts should be run.

**Hint:** Look for scripts that end in **dm** if you are trying to determine how the display manager is started.

You may find that the script for running a display manager, say /etc/init.d/rc5.d/S16xdm, may be a small script that contains additional logic to determine which display manager will really be run. So, while many systems allow these to be controlled through configuration, you can also find out what display manager will run by examining your initialization files.

It should come as no surprise that you can control whether your display manager is started at system startup simply by creating symbolic links for starting and stopping it in the appropriate rc*n.d* directory. Furthermore, if you need to stop or start the display manager, you can use the script from /etc/init.d directly as shown in Listing 12.

**Listing 12. Stopping and starting a display manager**

```
root@pinguino:~# /etc/init.d/gdm stop
 * Stopping GNOME Display Manager...                                [ ok ]
root@pinguino:~# /etc/init.d/gdm start
 * Starting GNOME Display Manager...                                [ ok ]
```

Now that you know how to control starting and stopping a display manager, let's look at configuring each of our three display managers.

## XDM

The X Display Manager (XDM) is included in the XFree86 and X.Org packages. Under the Filesystem Hierarchy Standard, the configuration files should be located in /etc/X11/xdm. The main configuration file is /etc/X11/xdm/xdm-config. This file contains the location of other files used by XDM, information on authorization requirements, the names of scripts run to perform the various tasks for a user, and some other configuration information.

The Xservers file determines which local display or displays should be managed by XDM. It usually contains a single line as shown in Listing 13.

**Listing 13. Sample Xservers file**

```
:0 local /usr/X11R6/bin/X :0 vt07
```

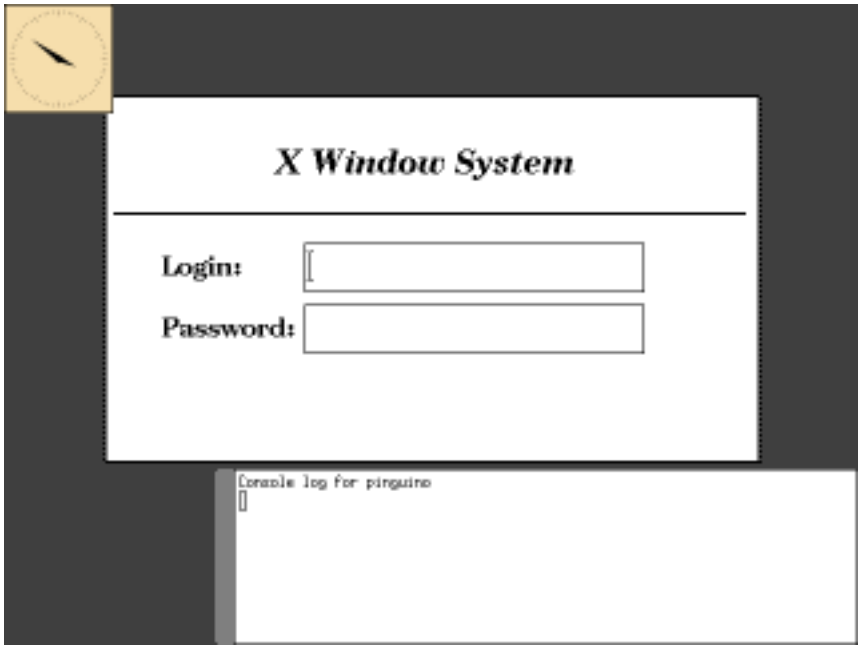
Listing 13 indicates that X should be run on virtual terminal 7. Most systems support using Ctrl-Alt-F1 through Ctrl-Alt-F7 to switch between virtual terminals, where vt01 through vt06 are text-mode terminals, and vt07 is the X terminal.

If you wish to support remote X terminals, then you need an Xaccess file. This file controls how XDM communicates with terminals that support the *X Display Manager Control Protocol* (*XDCMP*). Terminals that do not support this protocol are defined in

the Xservers file. XDCMP uses the well-known UDP port 177. For security reasons, you should restrict XDCMP use to a trusted internal network with suitable firewall protection.

You can customize the way XDM works by updating the scripts in /etc/X11/xdm. In particular, the Xsetup (or Xsetup_0) script lets you customize the greeting. Figure 2 shows a simple XDM greeting with a digital clock added.

**Figure 2. A modified XDM greeting**




The source for the modified Xsetup_0 file is shown in Listing 14.

**Listing 14. Sample Xsetup_0 file**

```
#!/bin/sh
xclock -geometry 80x80 -bg wheat&
xconsole -geometry 480x130-0-0 -daemon -notify -verbose -fn fixed -exitOnFail
```

The greeting shown in Figure 2 came from a system using a 640x480 pixel screen resolution at 256 colors. XDM uses the default resolution from your XF86Config or xorg.conf file. To change your default system-wide screen resolution, you may edit this file or use the utilities that may have come with your system. Listing 15 shows the Screen section of an XF86Config file. Note that the DefaultDepth is 16, so the X server will try to run the screen at the first possible resolution specified for this depth, 1024x768 in this case.

**Listing 15. Configuring screen resolution**

```
Section "Screen"
  DefaultDepth 16
  SubSection "Display"
    Depth      15
    Modes      "1280x1024" "1024x768" "800x600" "640x480"
  EndSubSection
  SubSection "Display"
    Depth      16
    Modes      "1024x768" "800x600" "640x480"
  EndSubSection
  SubSection "Display"
    Depth      24
    Modes      "1280x1024" "1024x768" "800x600" "640x480"
  EndSubSection
  SubSection "Display"
    Depth      32
    Modes      "1280x1024" "1024x768" "800x600" "640x480"
  EndSubSection
  SubSection "Display"
    Depth      8
    Modes      "1280x1024" "1024x768" "800x600" "640x480"
  EndSubSection
  Device       "Device[0]"
  Identifier   "Screen[0]"
  Monitor      "Monitor[0]"
EndSection
```

Note that `Depth` refers to the number of bits that make up each pixel. You may also see this called *bits per pixel* or *bitplanes*. Thus, using 8 bitplanes or 8 bits for each color allows up to 256 colors, while a depth of 16 allows up to 65536 colors. With today's graphic cards, the higher depths of 24 and 32 are now common.

You can check the screen resolution using the `xwininfo` command with the `-root` option to see the characteristics of your running X server, as shown in Listing 16.

**Listing 16. Checking the screen resolution**

```
ian@lyrebird:~> xwininfo -display 0:0 -root

xwininfo: Window id: 0x36 (the root window) (has no name)

  Absolute upper-left X:  0
  Absolute upper-left Y:  0
  Relative upper-left X:  0
  Relative upper-left Y:  0
  Width: 1024
  Height: 768
  Depth: 16
  Visual Class: TrueColor
  Border width: 0
  Class: InputOutput
  Colormap: 0x20 (installed)
  Bit Gravity State: NorthWestGravity
  Window Gravity State: NorthWestGravity
  Backing Store State: NotUseful
  Save Under State: no
  Map State: IsViewable
  Override Redirect State: no
  Corners:  +0+0  -0+0  -0-0  +0-0
  -geometry 1024x768+0+0
```

## KDM

KDM is the K Desktop Manager for the K Desktop Environment (KDE). KDE version 3 uses a kdmrc configuration file, a change from earlier versions, which used configuration information that was based on the xdm configuration files. This is located in the $KDEDIR/share/config/kdm/ directory, where $KDEDIR might be /etc/kde3/kdm/ or perhaps somewhere else. For example, on a SUSE SLES8 system, this is located in /etc/opt/kde3/share/config/kdm.

**Listing 17. KDM configuration file - kdmrc**

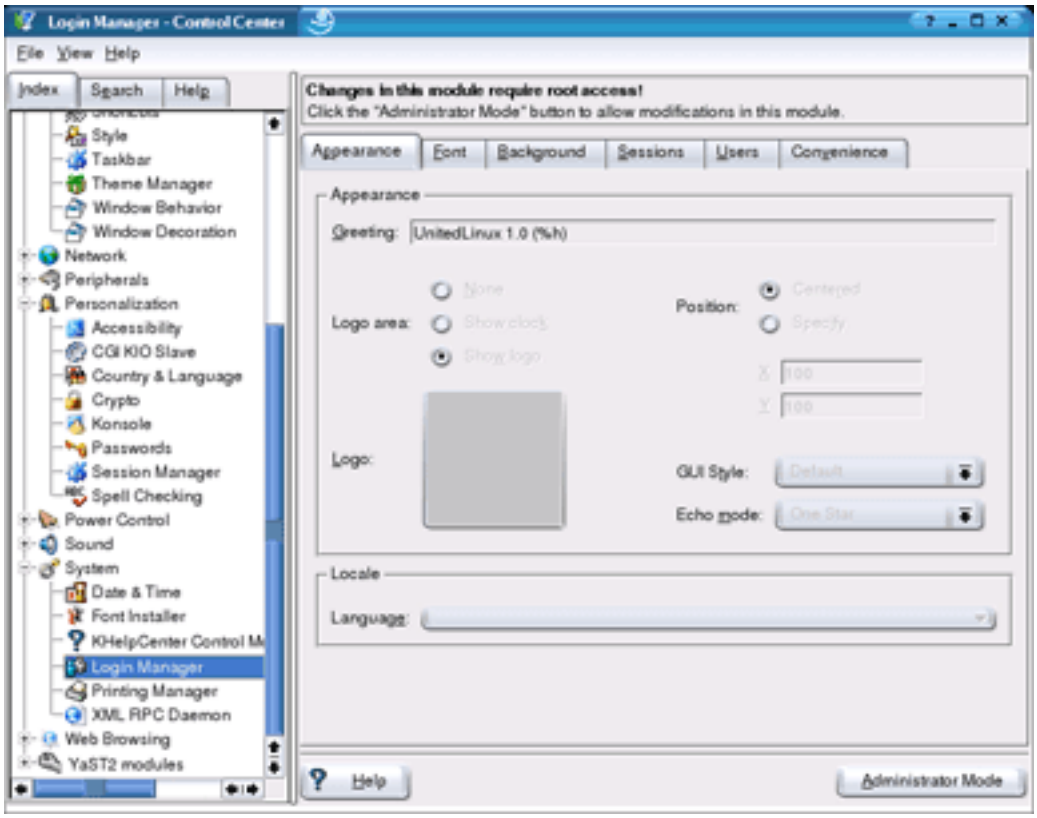
```
[Desktop0]
BackgroundMode=VerticalGradient
Color1=205,205,205
Color2=129,129,129
MultiWallpaperMode=NoMulti
Wallpaper=UnitedLinux-background.jpeg
WallpaperMode=Scaled

[X-*-Greeter]
GreetString=UnitedLinux 1.0 (%h)
EchoMode=OneStar
HiddenUsers=nobody,
BackgroundCfg=/etc/opt/kde3/share/config/kdm/kdmrc
MinShowUID=500
SessionTypes=kde,gnome,twm,failsafe

[General]
PidFile=/var/run/kdm.pid
Xservers=/etc/opt/kde3/share/config/kdm/Xservers

[Shutdown]
HaltCmd=/sbin/halt
LiloCmd=/sbin/lilo
LiloMap=/boot/map
RebootCmd=/sbin/reboot
UseLilo=false

[X-*-Core]
Reset=/etc/X11/xdm/Xreset
Session=/etc/X11/xdm/Xsession
Setup=/opt/kde3/share/config/kdm/Xsetup
Startup=/etc/X11/xdm/Xstartup
AllowShutdown=Root

[Xdmcp]
Willing=/etc/X11/xdm/Xwilling
Xaccess=/etc/X11/xdm/Xaccess
```

Many sections contain the same type of configuration information as for XDM, but there are some differences. For example, the SessionTypes field allows KDM to start any one of several different session types; other commands allow KDM to shut down or reboot the system.

You can configure KDM by editing the kdmrc file. You can also change many of the Login Manager settings by using the KDE control center (kcontrol) as shown in

Figure 3.

**Figure 3. Modifying KDM configuration with kcontrol**




The KDM handbook (see Resources) contains extensive information on KDM configuration.

## GDM

GDM is the GNOME Desktop Manager for the GNOME Desktop Environment. This desktop manager was not based on XDM, but was written from scratch. GDM uses a gdm.conf configuration file, normally located in the /etc/X11/gdm directory. Listing 18 shows part of a gdm.conf file.

**Listing 18. Partial GDM configuration file - gdm.conf**

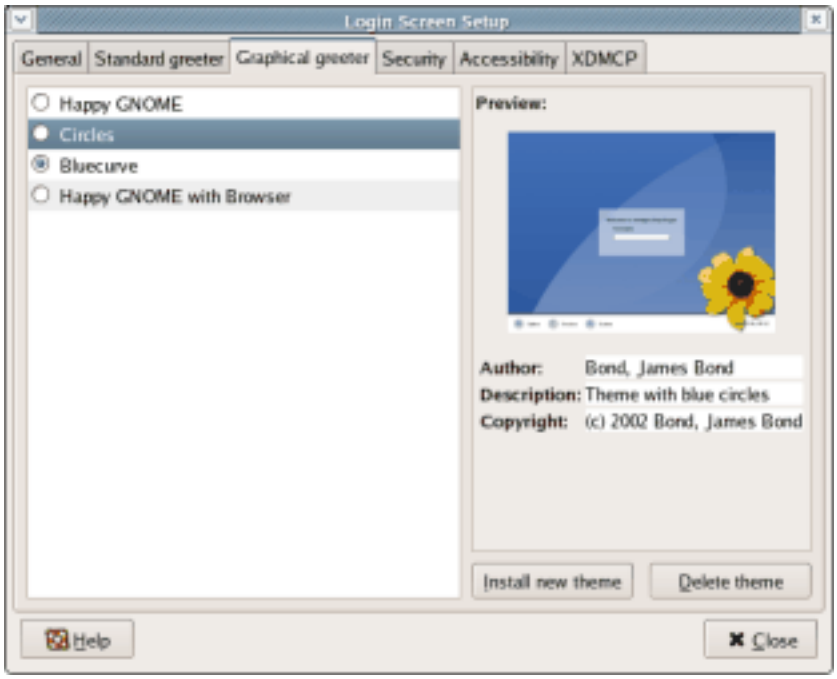
```
# You should probably never change this value unless you have a weird setup
PidFile=/var/run/gdm.pid
# Note that a post login script is run before a PreSession script.
# It is run after the login is successful and before any setup is
# run on behalf of the user
PostLoginScriptDir=/etc/X11/gdm/PostLogin/
PreSessionScriptDir=/etc/X11/gdm/PreSession/
PostSessionScriptDir=/etc/X11/gdm/PostSession/
DisplayInitDir=/etc/X11/gdm/Init
   ...
```

```
# Probably should not touch the below this is the standard setup
ServAuthDir=/var/gdm
# This is our standard startup script.  A bit different from a normal
# X session, but it shares a lot of stuff with that.  See the provided
# default for more information.
BaseXsession=/etc/X11/xdm/Xsession
# This is a directory where .desktop files describing the sessions live
# It is really a PATH style variable since 2.4.4.2 to allow actual
# interoperability with KDM.  Note that <sysconfdir>/dm/Sessions is there
# for backwards compatibility reasons with 2.4.4.x
#SessionDesktopDir=/etc/X11/sessions/:/etc/X11/dm/Sessions/:/usr/share/gdm/Buil\
tInSessions/:/usr/share/xsessions/
# This is the default .desktop session.  One of the ones in SessionDesktopDir
DefaultSession=default.desktop
```

Again, you will see some similarities in the type of configuration information used for GDM, KDM, and XDM, although gdm.conf is a much larger file with many more options.

You can configure GDM by editing the gdm.conf file. You can also change many of the settings by using the `gdmsetup` command. Figure 4 shows an alternate graphical greeter available on a Fedora system.

**Figure 4. Modifying GDM configuration with gdmsetup**




The GNOME Display Manager Reference Manual (available from the gdmsetup help, or see Resources) contains extensive information on GDM configuration.

# Section 4. Customize a window manager

This section covers material for topic 1.110.4 for the Junior Level Administration (LPIC-1) exam 101. The topic has a weight of 5.

In this section, you learn how to:

- Customize a system-wide desktop environment or window manager
- Customize window manager menus and desktop panel menus
- Configure an X terminal
- Verify and resolve library dependency issues for X applications
- Export an X display

## Window managers

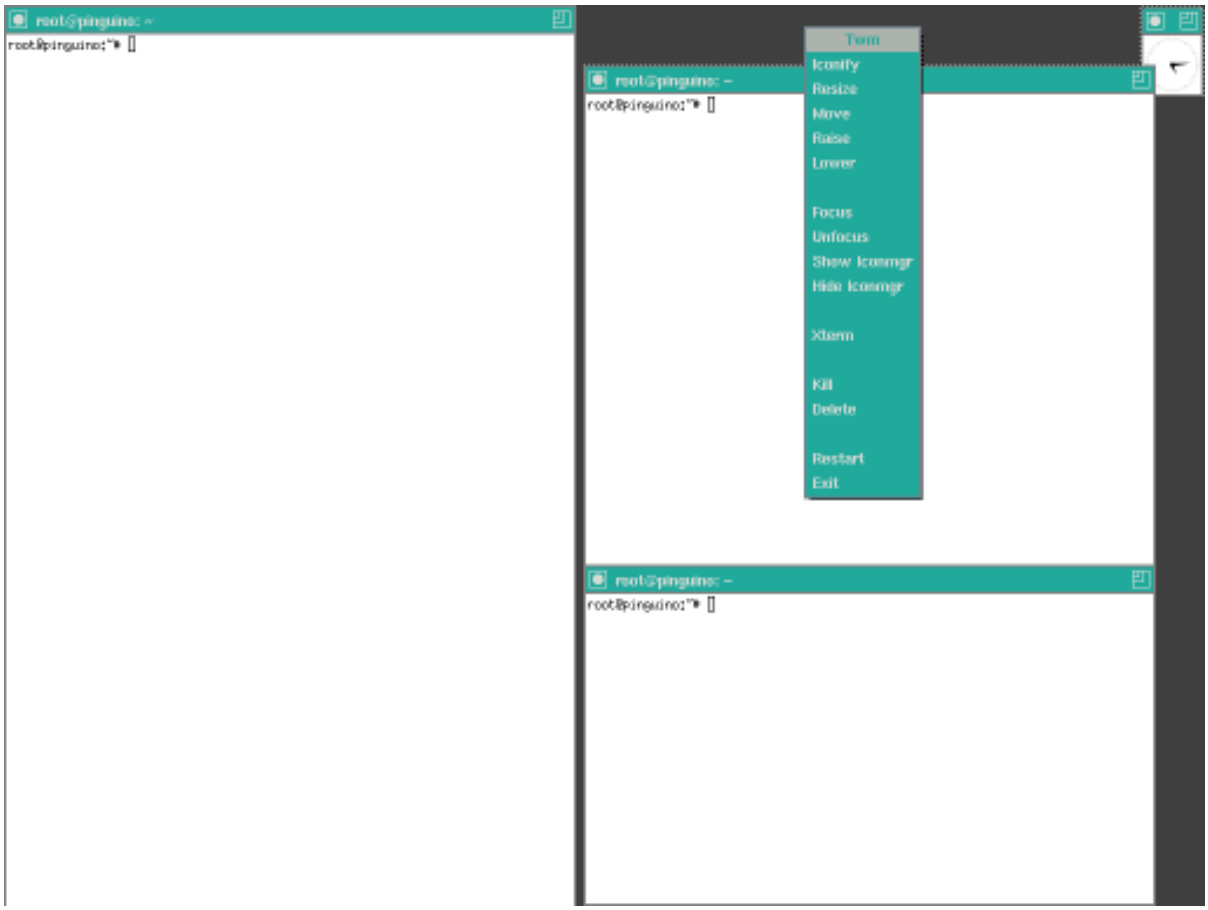
In the previous section, you learned about display managers and how to set them up. You also learned in this tutorial that while X is a toolkit enabling applications to create graphical windows, it does not specify a user interface. In this section, you learn more about user interfaces and how to configure what happens after you have an X session up and running.

You can imagine that, without any user interface specification, developer creativity might result in many different styles of windows, all vying for space on your screen, and all with different keystrokes, mouse operations, and styles for things like buttons, dialog boxes, and so on. To bring some order to chaos, higher level toolkits were developed. These in turn gave rise to *window managers*, such as twm, fvwm, and fvwm2 and finally to desktops such as KDE and GNOME.

Desktops provide a consistent user experience but also consume considerable CPU and memory resources. Before computers had the power to support the likes of a KDE or GNOME desktop, window managers were popular, and many users still like them for their light weight and fast response.

If you have just installed X and you type in the command `startx`, then you will see a display something like Figure 5.

**Figure 5. Running twm with startx**




This is the twm window manager, shown with the menu that is obtained by pressing mouse button 1 (normally the left button for right-handed users) over the background. You will notice three terminal windows and an analog clock, but no task bars, launchers, or other desktop paraphernalia.

The `startx` command is actually a front end command to `xinit`, which starts the X server process and some client applications. It is usually located in /usr/X11R6/bin, as is `xinit` and many other X utilities. X applications can take settings from an X resources database as well as a command line. Table 6 summarizes the names and purpose of each of the configuration files used by either `startx` or `xinit`. Note that some or all of these files may not be present on a particular system or user home directory.

Table 6. Configuration files for startx and xinit

| File | Description |
|---|---|
| $HOME/.xinitrc | User-defined executable script that merges in resource files and starts client applications |
| $HOME/.xserverrc | User-defined executable script that provides overrides for default X |

| | |
|---|---|
| | server configuration |
| /usr/X11R6/lib/X11/xinit/xinitrc | System default executable script that merges in resource files and starts client applications |
| /usr/X11R6/lib/X11/xinit/xserverrc | System default executable script that provides overrides for default X server configuration |
| $HOME/.Xresources | User-defined file describing resources for X applications |
| $HOME/.Xmodmap | User-defined file defining keyboard and mouse settings |
| /usr/X11R6/lib/X11/xinit/.Xresources | System default file describing resources for X applications |
| /usr/X11R6/lib/X11/xinit/.Xmodmap | System default file defining keyboard and mouse settings |

Note carefully that the system xinitrc and xserverrc file omit the leading dot, while all the others have it.

Every window on a screen, and indeed every widget on the screen, has attributes such as height, width, and placement (geometry), foreground and background colors or images, title text and color, and so on. For a new client application, most of these values can be supplied on the command line. Since there are many attributes, it is easier to have defaults. Such defaults are stored in a *resource database*, which is built from resource files using the `xrdb` command.

Listing 19 shows the default xinit file shipped with XFree86 4.5.0.

**Listing 19. Sample xinit file - /usr/X11R6/lib/X11/xinit/xinitrc**

```
#!/bin/sh
# $Xorg: xinitrc.cpp,v 1.3 2000/08/17 19:54:30 cpqbld Exp $

userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/usr/X11R6/lib/X11/xinit/.Xresources
sysmodmap=/usr/X11R6/lib/X11/xinit/.Xmodmap

# merge in defaults and keymaps

if [ -f $sysresources ]; then
    xrdb -merge $sysresources
fi

if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi

if [ -f $userresources ]; then
    xrdb -merge $userresources
fi
```

```
if [ -f $usermodmap ]; then
    xmodmap $usermodmap
fi

# start some nice programs

twm &
xclock -geometry 50x50-1+1 &
xterm -geometry 80x50+494+51 &
xterm -geometry 80x20+494-0 &
exec xterm -geometry 80x66+0+0 -name login
```

Note that the `xrdb`command is used to merge in the resources, and the xmodmap is used to update the keyboard and mouse definitions. Finally, several programs are started in the background with a final program started in the foreground using the `exec` command, which terminates the current script (`xinitrc`) and transfers control to the xterm window with geometry 80x66+0+0 . This is the login window, and terminating it will terminate the X server. There must be one such application, although some people prefer the window manager (twm in this example) to have this role. All other applications should be started in the background so the script can complete.

The first two values in the geometry specification define the size of the window. For a clock, this is in pixels, while for the xterm windows, it is in lines and columns. If present, the next two values define the placement of the window. If the first value is positive, the window is placed relative to the left edge of the screen, while a negative value causes placement relative to the right edge. Similarly, positive and negative second values refer to the top and bottom of the screen respectively.

Suppose you want your clock to be larger with a different colored face and to be located in the bottom right of the screen instead of the top right. If you just want this for one user, copy the above file to the user's home directory as .xinitrc (remember the dot) and edit the clock specification as shown in Listing 20. You can find all the color names in the rgb.txt file in your X installation directory tree (for example, /usr/X11R6/lib/X11/rgb.txt).

**Listing 20. Modifying xclock startup in xinitrc**

```
xclock -background mistyrose -geometry 100x100-1-1 &
```

If you want to update the defaults for the whole installation, you should update the /usr/X11R6/lib/X11/xinit/.Xresources and /usr/X11R6/lib/X11/xinit/.Xmodmap files instead of the dot versions for individual users.

Several tools can help you customize windows and keystrokes.

**xrdb**

Merges resources from a resource file into the X resource database for the

running X server. By default it passes the source resource file through a C++ compiler. Specify the `-nocpp` option if you do not have a compiler installed.

**xmodmap**
Sets keyboard and mouse bindings. For example, you can switch settings for left-handed mouse usage, or set up the backspace and delete keys to work the way you are accustomed to.

**xwininfo**
Tells you information about a window, including geometry information.

**editres**
Lets you customize the resources for windows on your screen, see the changes, and save them in a file that you can later use with xrdb.

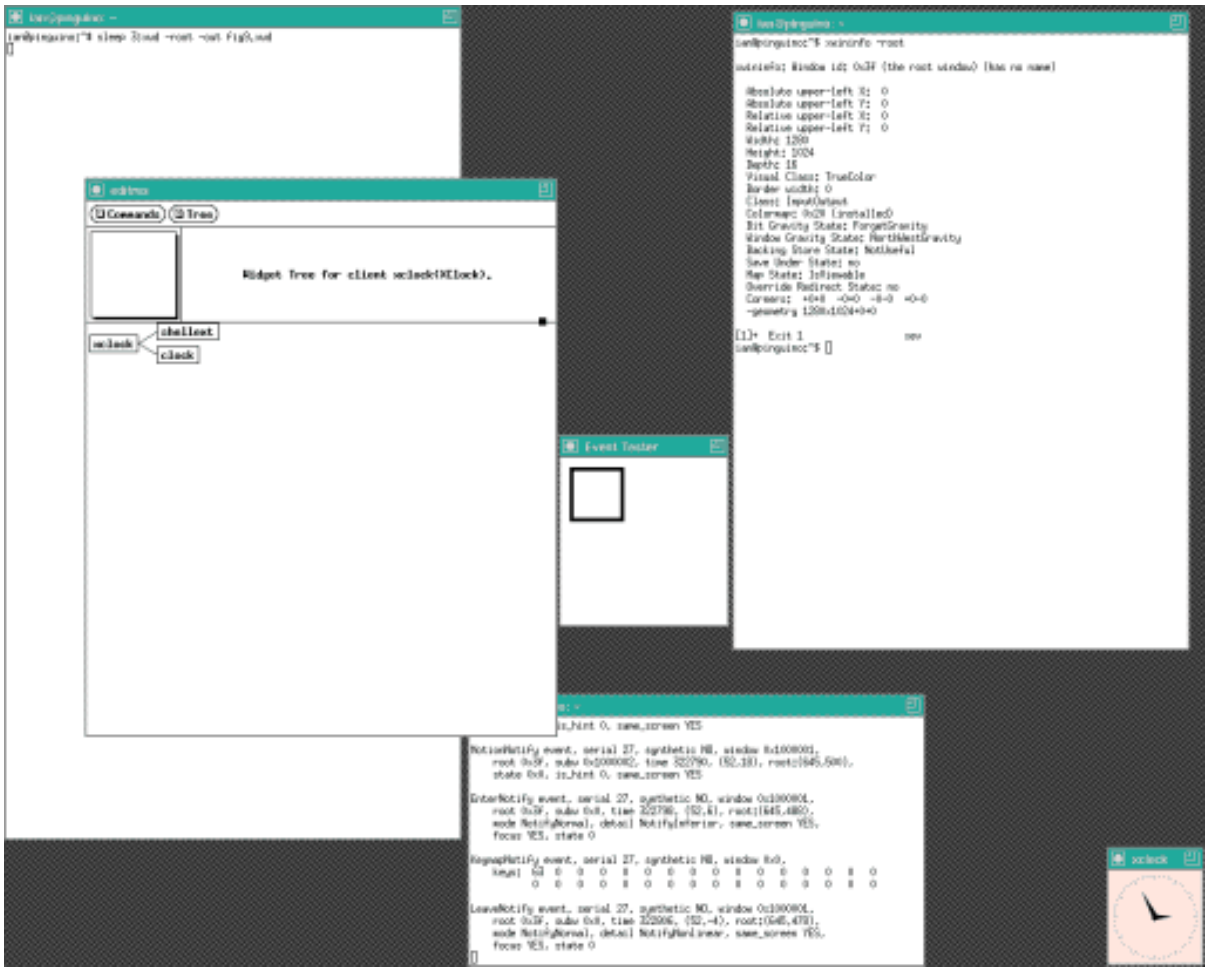
**xev**
Starts a window and captures X events that are displayed in the originating xterm window. Use this to help determine the appropriate codes to use when customizing keyboards or checking mouse events.

Use the man pages to find out more about each of these commands.

Figure 6 shows a busy screen with several of these commands running.

- The upper left terminal window (the login window) is using `xwd` to capture the whole screen to a file.
- The window overlapping it is running `editres` to modify the resources for the clock window.
- The small central window is running `xev`, and the output is going to the terminal window below. The right-hand window shows the output of `xwininfo` for the root window (the whole screen).
- The customized clock with its face colored mistyrose is running in the bottom right corner of the screen.

**Figure 6. Window information and configuration**



Besides the windows themselves, the window manager also allows customization. For example, the menu that was shown in Figure 5 is configured in a twm customization file. The system default is in the X installation directory tree (/usr/X11R6/lib/X11/twm/system.twmrc), and individual users may have a .twmrc file. If a user has multiple displays, there can be files (such as .twmrc.0 or .twmrc.1) for each display number. Listing 21 shows the part of the system.twmrc file that defines the menu shown in Figure 5.

### Listing 21. Menu configuration in twm

```
menu "defops"
{
"Twm"   f.title
"Iconify"       f.iconify
"Resize"        f.resize
"Move"          f.move
"Raise"         f.raise
"Lower"         f.lower
""              f.nop
"Focus"         f.focus
"Unfocus"       f.unfocus
"Show Iconmgr"  f.showiconmgr
```

```
"Hide Iconmgr"  f.hideiconmgr
""              f.nop
"Xterm"         f.exec "exec xterm &"
""              f.nop
"Kill"          f.destroy
"Delete"        f.delete
""              f.nop
"Restart"       f.restart
"Exit"          f.quit
}
```

See the man page for twm or your preferred window manager for more information.

## Desktops

If you are using a display manager or desktop, you will find that these also can be configured. Indeed, you already saw the Xsetup_0 file for XDM in the previous section. As with the window manager configuration that you have just seen, desktop configuration can be done system-wide or per-user.

### GNOME customization

GNOME is configured mostly using XML files. System defaults are found in directories in the /etc filesystem, such as /etc/gconf, /etc/gnome, and /etc/gnome-vfs2..0, along with other directories for specific GNOME applications. User configuration is usually found in subdirectories of the home directory that start with .g. Listing 22 shows several of the possible locations for GNOME configuration information.

**Listing 22. GNOME configuration locations**

```
[ian@lyrebird ian]$ ls -d /etc/g[cn]*
/etc/gconf  /etc/gnome  /etc/gnome-vfs-2.0  /etc/gnome-vfs-mime-magic
[ian@lyrebird ian]$ find . -maxdepth 1 -type d -name ".g[nc]*"
./.gnome2
./.gconfd
./.gconf
./.gnome
./.gnome2_private
./.gnome-desktop
./.gnome_private
```

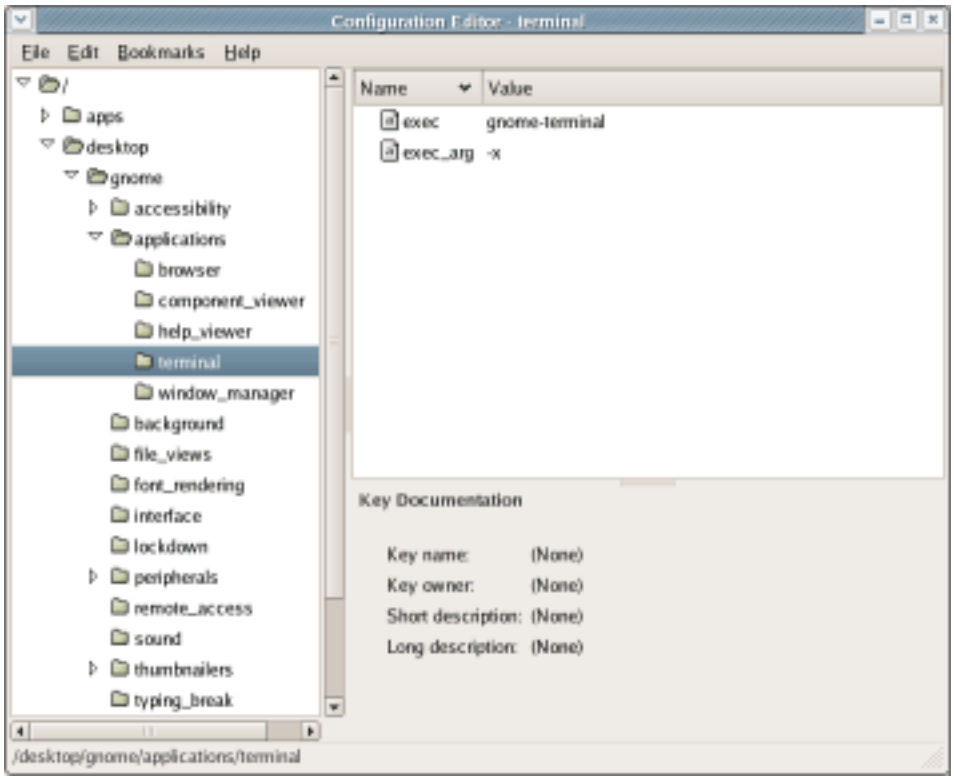
Rather than extensive man pages, GNOME has an online manual; you can access the manual from the `gnome-help` command, or from a menu item such as Desktop > Help. As of this writing, the manual has three major sections: Desktop, Applications, and Other Documentation. The table of contents for a recent version of the Desktop Help is shown in Figure 7.

**Figure 7. Window information and configuration**


You will find information about configuration tools under the System Administration Guide in the Desktop section and also under the Configuration Editor Manual in the applications topic in the Desktop section.

You may launch the graphical configuration editor using the `gconf-editor` command or by selecting Configuration Editor from the Applications > System Tools menu. The configuration for the gnome-terminal application is shown in Figure 8.

**Figure 8. Window information and configuration**




In addition to graphical tools, there is also a `gconftool-2` command line program for interrogating and updating GNOME configuration information. See the above mentioned System Administration Guide for more details.

### KDE customization

KDE is configured using plain text files with UTF-8 encoding for non-ASCII characters. As with GNOME, there may be many different configuration files. If multiple identically named files exist in different parts of the configuration tree, then the values from these are merged. Sy6stem values are located in the $KDEDIR/share/config tree, where $KDEDIR might be /etc/kde3/kdm/ or perhaps somewhere else. For example, on a SUSE SLES8 system, this is located in /etc/opt/kde3/share/config. User-specific customizations are located in the .kde/share tree in the user's home directory.

Configuration files have one or more group names enclosed in square brackets, followed by key and value pairs. The key may contain spaces and is separated from the value by an equal sign. The configuration file for the konqueror browser is shown in Listing 23.

### Listing 23. KDE configuration file for konqueror browser

```
[HTML Settings]
[Java/JavaScript Settings]
```

```
ECMADomainSettings=localhost::Accept
JavaPath=/usr/lib/java2/jre/bin/java
EnableJava=true
EnableJavaScript=true

[EmbedSettings]
embed-text=true
embed-audio=false
embed-video=false

[Reusing]
MaxPreloadCount=1
PreloadOnStartup=true
```

The configuration files may be edited manually. Most systems will include a graphical editing tool, such as KConfigEditor or a tool customized for the distribution, such as the SUSE Control Center.

## Different xterms

The usual `xterm` program that is installed with graphical desktops is very functional, but also uses quite a lot of system resource. If you are running a system with many X terminal clients running on a single processor, you may want to consider using a lighter-weight terminal. Two such are `rxvt` and `aterm` (which was built on top of `rxvt`). These are VT102 emulators, which are not usually installed by default, so you will need to install them.

## Required libraries

By now you may be realizing that there are many different libraries and toolkits that are used for X applications. So how do you ensure that you have the right libraries? The `ldd` command lists library dependencies for any application. In its simplest form, it takes the name of an executable and prints out the libraries required to run the program. Note that `ldd` does not automatically search your PATH, so you usually need to give the path (relative or absolute) as well as the program name, unless the program is in the current directory. Listing 24 shows the library dependencies for the three terminal emulators that you saw above. The number of library dependencies for each gives you a clue about the relative system requirements of each.

**Listing 24. Library dependencies for xterm, aterm, and rxvt**

```
root@pinguino:~# ldd `which xterm`
    linux-gate.so.1 =>  (0xffffe000)
    libXft.so.2 => /usr/X11R6/lib/libXft.so.2 (0xb7fab000)
    libfontconfig.so.1 => /usr/X11R6/lib/libfontconfig.so.1 (0xb7f88000)
    libfreetype.so.6 => /usr/X11R6/lib/libfreetype.so.6 (0xb7f22000)
    libexpat.so.0 => /usr/X11R6/lib/libexpat.so.0 (0xb7f06000)
    libXrender.so.1 => /usr/X11R6/lib/libXrender.so.1 (0xb7eff000)
```

```
        libXaw.so.7 => /usr/X11R6/lib/libXaw.so.7 (0xb7ead000)
        libXmu.so.6 => /usr/X11R6/lib/libXmu.so.6 (0xb7e99000)
        libXt.so.6 => /usr/X11R6/lib/libXt.so.6 (0xb7e4f000)
        libSM.so.6 => /usr/X11R6/lib/libSM.so.6 (0xb7e46000)
        libICE.so.6 => /usr/X11R6/lib/libICE.so.6 (0xb7e30000)
        libXpm.so.4 => /usr/X11R6/lib/libXpm.so.4 (0xb7e22000)
        libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0xb7e15000)
        libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7d56000)
        libncurses.so.5 => /lib/libncurses.so.5 (0xb7d15000)
        libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7be6000)
        libdl.so.2 => /lib/tls/i686/cmov/libdl.so.2 (0xb7be3000)
        /lib/ld-linux.so.2 (0xb7fc3000)
root@pinguino:~# ldd `which aterm`
        linux-gate.so.1 =>  (0xffffe000)
        libXpm.so.4 => /usr/X11R6/lib/libXpm.so.4 (0xb7f81000)
        libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7ec1000)
        libSM.so.6 => /usr/X11R6/lib/libSM.so.6 (0xb7eb9000)
        libICE.so.6 => /usr/X11R6/lib/libICE.so.6 (0xb7ea3000)
        libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7d75000)
        libdl.so.2 => /lib/tls/i686/cmov/libdl.so.2 (0xb7d72000)
        /lib/ld-linux.so.2 (0x80000000)
root@pinguino:~# ldd `which rxvt`
        linux-gate.so.1 =>  (0xffffe000)
        libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0xb7eb0000)
        libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7d81000)
        libdl.so.2 => /lib/tls/i686/cmov/libdl.so.2 (0xb7d7e000)
        /lib/ld-linux.so.2 (0x80000000)
```

## Exporting a display

An X display is known by a name of the form *hostname:displaynumber.screennumber*. For Linux running on a workstation such as a PC, there is typically only one display with a single screen. In this case, the *displayname* may be, and usually is, omitted so the display is known as :0.0. The DISPLAY environment variable is usually set to the display name., so you can display it using the command `echo $DISPLAY`. Depending on your system, this variable may or may not be set if you use `su -` to switch to another user. In such a case, you may need to set and export the DISPLAY as shown in Listing 25. In this listing you see an attempt to start the xclock application after switching to root, but the attempt fails because the DISPLAY environment variable is not set. Even if the DISPLAY variable is set, you still may not be able to use the display, as you will also need authorization to do so.

**Listing 25. Attempting to start xclock**

```
ian@lyrebird:~> whoami
ian
ian@lyrebird:~> echo $DISPLAY
:0.0
ian@lyrebird:~> su -
Password:
lyrebird:~ # echo $DISPLAY

lyrebird:~ # xclock
Error: Can't open display:
lyrebird:~ # export DISPLAY=:0.0
```

```
lyrebird:~ # echo $DISPLAY
:0.0
lyrebird:~ # xclock
Xlib: connection to ":0.0" refused by server
Xlib: No protocol specified

Error: Can't open display: :0.0
lyrebird:~ # export XAUTHORITY=~ian/.Xauthority
lyrebird:~ # xclock
lyrebird:~ # ls -l ~ian/.Xauthority
-rw-------  1 ian users 206 Feb 18 16:20 /home/ian/.Xauthority
```

Let's take a look at what is going on here. In this case, the user ian logged in to the system and his `DISPLAY` environment was set to :0.0 as we expect. When user ian switched to user root, the `DISPLAY` environment variable was not set, and an attempt to start `xclock` failed because the application did not know what display to use.

So the substituted user, root, set the `DISPLAY` environment variable, and exported it so that it would be available to other shells that might be started from this terminal window. Note that setting and exporting an environment variable does not use the leading $ sign, while displaying or otherwise using the value does. Note too, that if the `su` command had omitted the - (minus) sign, the `DISPLAY` environment variable would have been set as it had been for user ian. Nevertheless, even with the environment variable set, `xclock` still failed.

The reason for the second failure lies in the client/server nature of X. Although root is running in a window on the one and only display on this system, the display is actually owned by the user who logged in originally, ian in this case. Let's take a look at X authorization.

### Authorization methods

For local displays on Linux systems, authorization usually depends on a so-called **MIT-MAGIC-COOKIE-1**, which is usually regenerated every time the X server restarts. A user can *extract* the magic cookie from the .Xauthority file in his or her home directory (using the `xauth extract` command and give it to another user to *merge* into that user's .Xauthority file using the `xauth merge` command. Alternatively, a user can grant authority for other users to access the local system using the `xhost +local:` command.

### XAUTHORITY

Another alternative is to set the XAUTHORITY environment variable to the location of a file containing the appropriate MIT-MAGIC-COOKIE-1. When switching to root, it is easy to do this, since root can read files owned by other users. This is, in fact, what we did in Listing 25, so after setting and exporting XAUTHORITY to ~ian/.Xauthority, root is now able to open graphical windows on the desktop. We said we'd mention a difference with Red Hat systems. Using `su` to switch to root on

a Red Hat system is slightly different on a SUSE system, where the display setup is done automatically for you.

So what if you switch to another non-root user? You will notice from Listing 25 that the .Xauthority file for user ian allows only user read and write access. Even members of the same group cannot read it, which is what you want unless you'd like someone to open up an application that takes over your screen and prevents you from doing anything! So, if you do extract an MIT-MAGIC-COOKIE-1 from your .Xauthority file, then you have to find some secure way to give it to your friendly non-root user. Another approach is to use the `xhost` command to grant authority to any user on a particular host.

**The xhost command**

Because of the difficulty of securely passing an MIT-MAGIC-COOKIE-1 cookie to another user, you may find that, for a Linux system with a single user, `xhost` is easier to use, even though the `xauth` approach is generally preferred over the `xhost` command, However, keep in mind the network heritage of the X Window System so that you do not accidentally grant more access than you intended and thereby open up your system and allow arbitrary network users to open windows on your desktop.

To give all local users authority to open applications on the display (:0.0), user ian can use the `xhost` command. Open a terminal window on your desktop and enter this command:

```
xhost +local:
```

Note the trailing colon (:). This will allow other users on the same system to connect to the X server and open windows. Since you are a single-user system, this means that you can `su` to an arbitrary non-root user and can now launch `xclock` or other X applications.

You may also use `xhost` to authorize remote hosts. This is generally not a good idea except on restricted networks. You will also need to enable the appropriate ports on your firewall if you are using one.

Another alternative for using X applications from another system is to connect to the system using a *secure shell* (*ssh*) connection. If your default ssh client configuration does not enable X forwarding, you may need to add the `-X` parameter to your `ssh` command. The ssh server will also need to have X forwarding enabled. In general, this is a much more secure method of using X remotely than opening up your system to arbitrary connections using `xhost`.

For more details on using the `xauth` and `xhost` commands, you can use the commands `info xauth`, `man xauth`, `info xhost`, or `man xhost` as appropriate to view the online manual pages. If you are interested in security for X connections,

start with the manual pages for `Xsecure`.

# Resources

### Learn

- Review the entire LPI exam prep tutorial series on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the LPIC Program, find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- In "Basic tasks for new Linux developers" (developerWorks, February 2006), learn how to open a terminal window or shell prompt and much more.
- Get documentation for XFree86 version 4.5.0 and other versions on the XFree86 site.
- Get documentation for the X Window System Version 11 Release 6.9 and 7.0 on the X.Org site.
- The *X.Org Modular Tree Developer's Guide* will help you build the X.Org releases from source.
- An Xft tutorial describes the Xft font mechanism introduced in XFree86 4.0.2.
- kde.org is the home of the K Desktop Environment and KDE documentation, including *The kdm Handbook* and *KDE for System Administrators* .
- The GNOME Foundation is the home of the GNOME Desktop Environment and the *Gnome Display Manager Reference Manual* .
- The Linux Documentation Project has a variety of useful documents, especially its HOWTOs.
- *The Concise Guide to Xfree86 for Linux* (Que, 1999) provides more detail on installing, configuring and using X.
- *LPI Linux Certification in a Nutshell* (O'Reilly, 2001) and *LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 (Exam Cram 2)* (Que, 2004) are references for those who prefer book format.
- Find more tutorials for Linux developers in the developerWorks Linux zone.
- Stay current with developerWorks technical events and Webcasts.

### Get products and technologies

- Download XFree86 from The XFree86 Project, Inc.
- Download X.Org from The X.Org Foundation.
- Order the SEK for Linux, a two-DVD set containing the latest IBM trial software

for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

- Download IBM trial software directly from developerWorks.

**Discuss**

- Participate in the discussion forum for this content.
- Read developerWorks blogs, and get involved in the developerWorks community.

## About the author

Ian Shields
Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.