Insider Tips: The X Window System

MISTER X11

Thanks to automatic hardware detection, today's admins rarely need to configure the X window system manually. But if you want to use X11's excellent networking capabilities and tuning options, you will certainly benefit from some background knowledge.

BY MARC ANDRÉ SELIG

Linux continues to spread to more and more ex-Windows desktops. A major factor in the success of Linux is the availability of easy-to-use desktops and applications. This abundance of software, however, means that bona fide admins must spend more time setting up the underlying tools that handle mouse and keyboard controls, graphics, and screen output for those applications.

The X server is the component that underlies almost any graphical output on Linux. The major exceptions to this are the console and the SVGA-Lib [2] library, a legacy library that supports direct image output on Linux systems. The X server itself provides primitive routines for displaying windowing hierarchies. X server also provides simple hardware acceleration, as well as offering components that support keyboard and mouse handling.

If you do not feel at home with client/server architectures, the X protocol might confuse you at first, as the X11 server is often based on a small workstation underneath a user's desk, whereas the clients often reside on a fat machine in the data center.

Client and Server Swap Roles

In the topsy-turvy world of X, the server is a program that interacts directly with the user. It draws widgets (graphical elements) on the screen and accepts input. The clients use a server rather than drawing on the screen themselves. That is, a client will not typically need direct access to the graphics hardware, but will simply ask the server to perform graphical functions. Typical X clients include word processors, web browsers, or terminal emulators.

If the client and server run on separate machines, the X server is almost always a desktop machine. This is also the case with typical Linux-based thin clients, which include a complete X server. It doesn't really make much difference where the client is. Given a fast enough network connection, the client can reside in the data center next door, or on the other side of the world. This said, the client often resides on the same machine as the server.

A so-called kiosk system that runs just a single application – an Internet cafe, for instance, or a viewer for X ray images – is a good example of a useful application for an X server with a single client.

Two commands are all it takes to set up a simple kiosk system. The administrator first needs to launch the server and then the required client or clients. In the following command, the option *:3* stipulates that the server will run on the fourth (start counting at zero) local display (and not the fourth monitor):

```
X :3 < /dev/null > ➥
/dev/null 2>&1 &
exec firefox --display :3 &
```

The *xinit* program facilitates this syntax by taking care of launching the X server.

The following line launches the X server and then Xterm, unless the *.xinitrc* below the user's home directory has a different setting:

```
xinit -fn 9x13 -- :3 > ➥
/dev/null 2>&1 &
```

The */etc/X11/XF86Config-4* file is the core of every X environment. Depending on your X server variant and distribution, this file might be called *xorg.conf* or something similar. The file has the configuration for all devices that cooperate with the X server.

As Linux has made amazing progress with regard to hardware detection, administrators don't normally need to modify the server configuration, and if they do, the changes are minimal. Programs that automatically create this file are *xf86cfg*, *xf86config*, or for Suse, Sax or Sax 2.

The configuration file is not only specific to the computer and graphics adapter, but also to the monitor. Before you change your monitor or graphics adapter, you need to make sure you have an alternative method of accessing the computer, (via the network or the console), just in case the options are wrong and the X server will not launch.

Window Managers

Left to its own devices, the X server is not very user friendly, especially not if you need to work with multiple programs at the same time. A structuring component is needed to support more convenient use of the workstation. The window manager adds title bars to individual windows, to help identify them more easily, and a frame to allow users to scale windows. Figure 1 shows TWM, one of the oldest window managers.

TWM allows users to move and overlap windows, and even sorts them automatically. It also implements a simple menu that allows users to launch programs and iconize or scale windows. Legacy window mangers such as TWM or FVWM are lean and efficient, but still convenient. Today's thin clients or older

Figure 1: Although the TWM window manager provides only minimal functionality, it is easy to use and convenient.

machines use them to save space and resources.

Three lines are all you need to launch an X session with TWM and Firefox:

```
X :3 < /dev/null > ➥
/dev/null 2>&1 &
twm -display :3 &
exec firefox --display :3 &
```

Today, most users prefer a full-fledged desktop environment such as KDE or Gnome. These all-encompassing frameworks have tools, utilities, and extensive libraries in addition to a window manager. Libraries help to simplify GUI programming, giving programs a uniform look and feel, and optimizing communication between applications.

Modern desktop environments add a session manager to the window manager. The session manager restores the previous session, launching the programs used last and putting the windows where they belong. Both Gnome and KDE have powerful session managers.

GUI-based Login

The X server handles screen input and output, and the window manager facilitates the use of multiple windows. That really only leaves one wish on the user's list, the ability to log on to the system in a graphical window. There are not many users today who relish the thought of launching a GUI session using console commands.

A display manager is a convenient feature. It calls an X server and opens a dialog box like the one shown in Figure 2. After authenticating the user, the display manager automatically launches the configured environment. If needed, the X11 system's networking capabilities mean that the display manager doesn't even need to run on the same machine as the X server. The display manager launches like a simple client. The X server can then use the XDMCP protocol to talk to the display manager on another machine.

The administrator can either assign a computer as the display manager host, or the X server can broadcast packets across the LAN to automatically locate a suitable display manager. The command looks like this: *X -broadcast*. To launch the display manager on a dedicated host, you need to enter *X -query host* to launch the X server.

All modern Linux distributions now install a display manager and a matching configuration along with X11. The easiest way of finding out where these components are is to run *locate* [3], specifying the name of the manager component you are looking for. The legacy XDM and the alternative KDE (KDM) (Figure 2) and Gnome (GDM) components are widespread.

Controlling Displays

Computers are not restricted to launching a single X server. With its multiple virtual terminals, Linux can simultaneously support multiple servers. Special X software applications can even run with-

Figure 2: KDE's KDM display manager has a menu to allow users to select a desktop or shut down the system.

out a monitor and keyboard, which is important for terminal servers. Because the client and the server run independently and don't even need to reside on the same machine, it is important for the clients to know to which display to send their output. The *DISPLAY* environmental variable takes care of specifying a destination for the display data.

The variable contains an optional hostname, a display number, and an optional screen number: *(Host):Display(.Screen)*. The screen number is not typically needed, except by Xinerama environments where a single X server controls multiple displays. If the hostname is not specified in *DISPLAY*, clients automatically use the local machine, using Unix sockets rather than TCP ports to communicate (just like in an X11 network connection.)

When a user logs in via a display manager, the environmental variables are automatically set. To launch an X program automatically, you may first need to set a variable. The following syntax launches the Firefox browser on the local machine; the X server sends the Firefox output to a computer called *sun14*:

```
export DISPLAY=sun14:0
firefox &
```

If you are using X11 on a network, there are two important considerations you should keep in mind. First, servers do not accept connections from arbitrary clients – access by arbitrary clients would entail security risks. Second, the data stream from the X session is unencrypted by default. Attackers could easily sniff the protocols and hijack sessions.

Almost any modern X server applies access controls by default. There are two alternative systems for access controls: Xhost allows access to dedicated computers, which does not make it very secure; Xauth uses cryptographic cookies and thus supports very fine-grained controls.

These protection schemes are an important part of the X environment, as X clients can influence the server and other clients to a great extent using a variety of protocols. For example, keyboard logging is very simple. Make sure you only allow trusted programs to access your own X server.

Cookie-based Access Controls

The Xauth tool stores its cookies in a file called *.Xauthority* below the user's home directory. Whenever the user launches a client system, the client parses this *.Xauthority* file and uses the appropriate cookie to authenticate with the X server. If you are running a session on another machine, or if you are using different credentials on your own machine, you will need to import the cookie to your *.Xauthority* first (see Listing 1.) *xauth nextract* stores the cookie in a file, and *xauth nmerge* imports the file to *.Xauthority*.

The Xhost variant makes sense as an option on stand-alone machines to disable access controls completely. The *xhost +* command allows any client on any host to access the display. The *xhost -* command reenables access controls for the client.

Xauth only controls who is allowed to access the X server – it does not protect the data transmission at all. If you use the mechanism in Listing 1 to launch clients on remote machines, you should be aware of the danger of sniffing.

The popular SSH protocol helps tunnel the X protocol through an encrypted connection. The program provides encryption, sets the environment variables to the appropriate values, and uses Xauth to create an *.Xauthority* file on the remote machine.

Tunneling X over SSH

To set up a tunnel, you need to set the *X11Forwarding* variable in */etc/ssh/sshd_config*, and possibly in */etc/ssh/ssh_config*, to *yes*. Users can then set the *-X* command line flag to create a tunnel for X11 (Listing 2). SSH provides a virtual X server on the remote machine (typically *:10*); its clients use the display on the local X server. ■

Listing 1: Importing Session Cookies

```
01 #mas:~$ xauth nextract myxkey
   :0
02 #mas:~$ chmod 640 myxkey
03 #mas:~$ su unsafe
04 #Password:
05 #unsafe:/home/mas$ firefox
06 #Xlib: connection to ":0.0"
   refused by server
07 #Xlib: No protocol specified
08 #
09 #(firefox-bin:3086):
   Gtk-WARNING **: cannot open
   display: :0
10 #unsafe:/home/mas$ xauth
   nmerge myxkey
11 #unsafe:/home/mas$ firefox
```

Listing 2: X11 Tunnel over SSH

```
01 #mas@ishi:~$ ssh -X kanat.
   pair.com
02 #[...]
03 #mas@kanat:$ echo $DISPLAY
04 #localhost:10.0
05 #mas@kanat:$ firefox &
```

INFO

[1] X.org: *http://www.x.org/*

[2] SVGA-Lib: *http://www.svgalib.org*

[3] Marc André Selig, "Admin Workshop: Finding Files": Linux Magazine 02/05, p. 62.

THE AUTHOR

Marc André Selig currently works as a medical doctor in the Augsburg Regional hospital. He also does consultancy work for Linux, Solaris, and BSD installations. Recent work includes search engine technology and advanced full-text linking for scientific articles.

The Answers to your UNIX and Linux Systems Administration Questions, all in one place!

Sys Admin is all you need to keep your system up and running — ***Better than ever!***

- Security
- Storage Management
- Web
- System Monitoring
- Backup
- Networking
- Connectivity
- And more!

Sys Admin™

the journal for UNIX and Linux systems administrators

Devoted 100% to UNIX systems.

Get the only magazine devoted 100% to **UNIX** systems administration — solid, technical information full of ways to improve the performance and extend the capabilities of your system. Regular columns and departments also give you a solid look at important books, new product releases and upgrades, career opportunities, and technical meetings and conferences. Coverage spans a variety of platforms including Solaris, AIX, BSD, HP-UX, IRIX, SCO, Linux and others. If you administer a UNIX system – ***Sys Admin*** can save you and your organization time and money.

Where can you get more information? Visit our website at

www.**sysadmin**mag.com/sub/

Discount keycode: 2spd

There's only one place to get all the **UNIX** answers you and your organization need — ***Sys Admin!***