Module 18: X Windows in UNIX

Presently our communication with a computer is largely using a keyboard and a visual feedback from a display unit (VDU). To support human computer interaction (HCI), we need software support. For instance, we need a display in the form of a prompt on the screen. This reassures us on what is being input. This interface is further enhanced with iconic clues which help us launch the applications in place of character input.

In addition, many applications need a support to be able to display the results of a computation in a visual form. These require a software support in the form of drawing support on VDU. This entails support to be able to draw lines, join a set of lines in the form of continuous curve. It is even possible to generate a control panel to regulate the course of a set of operations in an application.

Unix provides the X-graphics library support to both OS and applications. In this chapter, we shall discuss the X-Windows support. For most Linux installations, we have a built-in X11 version library package. We begin with a brief description of what a Graphical User Interface (GUI) is. Later we briefly describe how the X-Windows system evolved.

18.1 Graphical User Interface (GUI)

GUI, as the name suggests, is an artifact that facilitates availing operating system facilities with the aid of widgets. The widgets basically are windows and icons which provide us with a clue to whatever we may be able to do.

One of the first well known GUIs was the Xerox STAR system which allowed users to have icons for a document page folded at a corner. The STAR had a printer icon and the operations were simply “drag and drop". Subsequently, these were adapted for the Apple Macintosh. Now-a-days all PCs and Workstations support a GUI. SunOS, the predecessor of Solaris had developed a GUI for Unix users which allowed users to have multiple terminals. Each open window is a terminal connected to the machine giving an illusion as if you, as a user, are multiply connected with the machine. Essentially, this gives users a capability similar to being connected to the machine from multiple terminals. Basically, it facilitates running many tasks at a time for a user. For instance, one may edit a program file in one window and run it from another window.

18.1.1 X-Window System

The X-Window system was developed in MIT under project Athena in 1984. The X-Window system operates on the basis of a client and server architecture. Further, X facilitates operation over a set of networked computers by defining a protocol. The protocol mechanism replaced function calls to let application draw its image in a window. The application is now the client and drawing the image is assigned to an X-server. The server handles the bit mapped graphics for the client application. In addition, the server also handles all display-related communications from the client. It is quite possible that for a single X-server there are many clients. In other words while handling communications from multiple clients, an X-server can in fact support drawing images for all connected clients. Clearly, this makes it possible to let many applications draw images on the same screen. The clients may be anywhere on a communication network. As almost all manufacturers support X-compliant protocols, we now have a capability to run an application on a machine from one manufacturer and display the image on another machine with a terminal from another manufacturer.

X-Servers and clients: X-servers support bitmapped displays and this happens as soon as a user logs onto a machine. Some systems may require that the user has to initiate the X-server operation by giving a *xinit* or a *xstart* command. On my set-up I use a *.xinitrc* run command file to customize my display terminal.

Usually X-clients are programs that communicate with one or more X-servers.

Window managers: The metaphor of desktop is important for a user. By analogy, a user may choose to activate one of the tasks from many on his desk. The window managers are designed to provide this illusion. In other words, a window manager (WM in short) must provide facilities that can support such desk-oriented operation from the view on the screen.

All operations within a window are supported by a window manager. The window screen is usually in the ratio of 3×4, i.e. it may be 600×800 pixels in size. It is possible to regulate this size. The write access into the window can be protected in a common desktop environment. If a user has been away and the system switches to the power-save mode then the system may restrict access only to a valid password holder. Alternatively, there may be a provision to lock a screen using the *xlock* command.

A window manager is a program, an X-client to be precise, which supports communication with keyboard, mouse and provides the basic interface for the user. A user can instruct the system to resize, iconify, delete a window or even magnify a part of the application image on the screen.

Though mostly the window manager runs on the same machine where the display is, it may actually reside on another machine!! The window manager maintains a “focus" which identifies the currently active window. This also helps to identify the application to which the data (or event) is to be communicated from keyboard (or mouse). Window focus may be obtained by clicking or simply by a “mouse-over" event. Window managers

also help to maintain top menu bars, support widgets like the pull-down menus to select an operation or to open a new window, *iconify*, resize, or delete (close) an existing window. So technically an X-server does no management, it only serves to display what the WM client asks it to display.

Some well known WMs: Some of the well known window managers are listed below:

1. *mwm* (Motif window manager)
2. *twm* (Tom's window manager). In *tvwm* v is for virtual.
3. *olwm* (Open look window manager). Sun's Open Look WM.

Hierarchy amongst windows: Practically every window manager supports widgets like menus (pull down and cascaded), dialog boxes with a variety of buttons like check button, radio button, selection from a list, or text input windows with scroll bars and alert windows to receive inputs like o.k. or cancel buttons. Usually, there is a root window and all the widgets are themselves drawn as small windows hierarchically within the parent window.

18.2 Some Standard X-clients

The Unix environment provides a slew of facilities in the form of clients that seek specific services. Let us briefly enumerate these clients and describe their functionalities:

- *xclock*: A clock display. It is possible to have a numerical or an analog display for the current time.
- *xbiff*: A Berkeley support program that displays mail status with a flag up, to show the arrival of a new mail.

- *xterm*: It provides a user with a new terminal window. With -C option the window can receive and display console messages.

Most of these clients have options like -geometry to indicate the size of display as shown in the example below.

xclock -geometry 80x80-10+10

This command seeks to display an 80£80 pixel clock, 10 pixels away from the right-hand corner of the screen. We can choose foreground and background presentation styles. One may also choose a title for a new window as shown below and even iconise it.

xterm -foreground green -background yellow -title "My Window"

xterm -iconic -title "My IconisedWindow"

One of the standard facilities all WMs provide is to cut and paste. It is usually possible to cut from any of the windows and paste it anywhere on the receiving window. This is very useful to transfer texts and images as bit maps.

18.3 Hosts

The X protocol requires to maintain a host-table in the /etc/hosts file. The machines listed in this table alone can set up a display on a host. For instance, my machine SE-0 has the following host table entries:

 # Internet host table

#

127.0.0.1 localhost

172.21.36.10 SE-0 loghost

210.163.147.1 a463-2.info.kochi-tech.ac.jp a463-2 canna-server loghost

210.163.147.2 main.info.kochi-tech.ac.jp a464-1 printer-server-host

172.21.36.20 SE-P0

172.21.36.2 SE-1

172.21.36.11 SE-2

.

.

172.21.36.17 SE-10

#

The host-table entries may be modified to add or delete hosts by using the *xhost* command as shown below.

To add an additional host:

xhost +addhostname

To delete an additional host:

xhost -deletehostname

(Caution: Not having a host name may delete all entries in the host-table)

18.4 Selecting a host for Display

With our ability to connect to a remote machine, we may run an application on a remote machine. Let us consider that we are executing the application on a host called *rem_host* while we are operating from a host loc host. However, if we wish to visualize the output on *loc_host* we shall have to choose the display to be available on loc host while the application may be generating the output for display on *rem_host.* This usually requires the use of X-protocol of display. We should have *loc_host* as well as *rem_host* identified in the host lists on both the machines. In this scenario it is required to set display on loc host. This is done as explained below. To begin with, let us interpret the following command:

xterm -display hostname:0.0

which will open a window on the hostname. The string :0.0 identifies the monitor and the *xserver* running on the hostname. Sometimes when we remote login using a telnet command, we may have to set the display from a remote computer to be on the host where we are presently logged in. This can be done by executing the following command in the window connecting to the remote machine.

setenv DISPLAY hostname:0.0

With this command executed on the remote machine we will get the application's image displayed on hostname. Note that the default, i.e. the local host is unix:0.0.

18.5 X-Utilities

Just as Unix system provides a command repertoire, X also has a suite of commands that provide some utilities. It is instructive to see the output for the following commands:

1. *showrgb*: Shows the colours that can be generated.
2. *xcalc*: Gives a hand hand calculator.

3. *xclipboard*: gives a clipboard to cut and paste.

4. *xclock*: Gives a clock running the time of the day.

5. *xfontsel*: Lists the fonts and font names.

6. *xhost*: Lists the hosts.

7. *xload*: Displays load on the system.

8. *xlsfonts*: Lists the fonts on the system.

9. *xmag*: Displays a magnified image of a part of the screen.

10 *xman*: Opens on line manual pages.

11 *xpr*: prints the image created by xwd.

12 *xrdb*: Loads resource setting in active database.

13 *xset*: Sets display, key board or mouse preferences.

14 *xterm*: emulate a character terminal.

15 *xwd*: stores a screen image.

16 *xwininfo*: Displays the window information.

The xrdb command has several options. Use it as follows:

xrdb -query .....this gives the resources defined for X-server.

xrdb -load .....use it to load new resource information (deleting old resource information).

xrdb -merge ..... merges new resource information with the existing.

xrdb -remove ..... removes the existing resource information.

(Note that .Xdefaults has the list of current resources.)

18.6 Startup

In this section we shall discuss the support offered for startup by .Xsessions, .Xdefaults files and .xinitrc files.

.Xsessions and .Xdefaults files: These files are usually located under X11 directory. On systems that employ xdm (the display manager) one can set up a sessions file as .Xsessions file to launch favorite applications on logging in. This may be with commands like:

xterm -sb & (For launching a terminal with a scroll bar)

xclock & (For launching a clock)

xterm -iconic (For launching an iconised terminal)

The .xdefaults file is utilized to select options for customization of applications display. For instance, one can choose the background colour for clock display, font size for character display and its colour by choosing an appropriate terminal foreground. A typical file may be as shown below:

```
! The exclamation sign identifies that this is a comment.
!
! Turn on the scroll bar
XTerm*scrollBar: True
! Select a font size
XTerm*Font: 10x20
! Lines to save on scroll
XTerm*saveLines: 100
! Mouse control to determine active window
Mvm*keyboardFocusPolicy: Pointer
```

One can load this file using the command

```
xrdb -load $HOME/.Xdefaults
```

.xinitrc files: Usually *xinit* starts the X-server on the host. The *xinit* command may be initiated from *.cshrc* (or .xinitrc or some other log in startup file) as one logs in. The .xinitrc may even launch a user's favourite applications like a browser, mail and a clock, etc. Some parts of my .xinitrc file are displayed below:

```
if [ -f ${HOME}/.Xdefaults ]; then
  xrdb -load ${HOME}/.Xdefaults
  else
    xrdb -load /usr/local/X11/lib/X11/Xdefaults
    fi
    echo `hostname` > ${HOME}/.display.machine
    echo `date +"seit: "%H.%M` >> ${HOME}/.display.machine
    chmod a+r $HOME/.display.machine
    echo `hostname` > ${HOME}/.display.host
    xclock -g 85x76+1064+743 -bw 0 &
    #xsetbg background3.gif
```

#xsetbg test2.gif

#xsetbg fujisun.gif

xsetbg marefoal.gif

/home/marco/bin/netscape/netscape -iconic&

xterm -g 80x10+0+0 -bw 5 -C -sb -bg MediumTurquoise -fg black +vb -fn 9x15 -iconic -fn fi xterm -g 80x10+50+50 -bw 5 -C -sb -bg MediumTurquoise -fg black +vb -fn 9x15 -iconic -fn xterm -g 80x20+210+210 -bw 5 -sb -bg SeaGreen -fg white -vb –fn 9x15 -iconic -fn fixed -T exec fvwm

~/clear_colormap -f /dev/cgsix0

logout

18.7 Motif and X

Motif seems to be currently the most favoured X-Windows interface on Unix-based systems. This helps in some respect, as all vendors tend to use the Motif library. Technically *mwm,* the Motif WM, is yet another application running on the X-server. The .mwmrc file can be used in these systems. Applications developed using the Motif library give a common look and feel just as the Windows does it for a PC.

Note that sometimes a user may have to copy the system's mwmrc file to user home directory to get the .mwmrc file. It is a good idea to take a peek at this file to see how the window (and its border), the mouse or arrow button roles bindings are defined. Below we show settings that one may find explanatory:

Buttons DefaultButtonBindings

{

<Btn1Down> icon|frame f.raise

<Btn2Down> icon|frame f.post_menu

<Btn2Down> root f.menu DefaultRootMenu

}

One may be able to customise a menu using f.exec as shown below. f.exec takes an argument which may run a X-utility.

Menu PrivateMenu

{

"Tools" f.title

```
"ManPages" f.exec "xman &"
"Cal" f.exec "xcalc &"
"Mail" f.exec "xterm -e Mail &"
}
```