Shobhit
Institute of Engineering & Technology
Deemed to-be-University
EDUCATION EMPOWERS

LINUX & X-WINDOWS PROGRAMMING

Prepared By:
AVINAV PATHAK
Assistant Professor
Shobhit Institute of Engineering and Technology
(Deemed to-be- University) MEERUT - 250110

CONTENTS

- Linux : An Introduction
- User Management in Linux
- Shells in Linux
- Using the X Windows System
- Handling the Events in X-Windows System

Linux O.S – An Introduction

In the simple language Linux is an operating system (OS). We all are familiar with other operating systems like Microsoft windows, Apple Mac OS, iOS, Google android, etc, just like them linux is also an operating system.

An operating system is a software that enables communication between computer hardware and software. It conveys input to get processed by the processor and brings output to the hardware to display it. This is the basic function of an operating system. Although, it performs many other important tasks, let's not talk about that.

Linux is around us since mid 90s. It can be used from wristwatches to supercomputers. It is everywhere in our phones, laptops, PCs, cars and even in refrigerators.It is very much famous among the developers and normal computer users.

Linux O.S – Historical Timeline

Evolution of Computer

In earlier days, computers were as big as houses or parks. So you can imagine how difficult it was to operate them. Moreover, every computer has a different operating system which made it completely worse to operate on them. Every software was designed for a specific purpose and was unable to operate on other computer. It was extremely costly and normal people neither can afford it nor can understand it.

Evolution of Unix

In 1969, a team of developers of Bell Labs started a project to make a common software for all the computers and named it as 'Unix'. It was simple and elegant, used 'C' language instead of assembly language and its code was recyclable. As it was recyclable, a part of its code now commonly called 'kernel' was used to develop the operating system and other functions and could be used on different systems. Also its source code was open source.

Initially, Unix was only found in large organizations like government, university, or larger financial corporations with mainframes and minicomputers (PC is a microcomputer).

Linux O.S – Historical Timeline

Unix Expansion

In eighties, many organizations like IBM, HP and dozen other companies started creating their own Unix. It result in a mess of Unix dialects. Then in 1983, Richard Stallman developed GNU project with the goal to make it freely available Unix like operating system and to be used by everyone. But his project failed in gaining popularity. Many other Unix like operating system came into existence but none of them was able to gain popularity.

Evolution of Linux

In 1991, Linus Torvalds a student at the university of Helsinki, Finland, thought to have a freely available academic version of Unix started writing its own code. Later this project became the Linux kernel. He wrote this program specially for his own PC as he wanted to use Unix 386 Intel computer but couldn't afford it. He did it on MINIX using GNU C compiler. GNU C compiler is still the main choice to compile Linux code but other compilers are also used like Intel C compiler.

Linux O.S – Historical Timeline

He started it just for fun but ended up with such a large project. Firstly he wanted to name it as 'Freax' but later it became 'Linux'.

He published the Linux kernel under his own license and was restricted to use as commercially. Linux uses most of its tools from GNU software and are under GNU copyright. In 1992, he released the kernel under GNU General Public License.

Linux Today

Today, supercomputers, smart phones, desktop, web servers, tablet, laptops and home appliances like washing machines, DVD players, routers, modems, cars, refrigerators, etc use Linux OS.

Linux O.S – Features / Characteristics

- **Multiuser capability:** Multiple users can access the same system resources like memory, hard disk, etc. But they have to use different terminals to operate.

- **Multitasking:** More than one function can be performed simultaneously by dividing the CPU time intelligently.

- **Portability:** Portability doesn't mean it is smaller in file size or can be carried in pen drives or memory cards. It means that it support different types of hardware.

- **Security:** It provides security in three ways namely authenticating (by assigning password and login ID), authorization (by assigning permission to read, write and execute) and encryption (converts file into an unreadable format).

- **Live CD/USB:** Almost all Linux distros provide live CD/USB so that users can run/try it without installing it.

ASSIGNMENT:

Ques:

1. What is Linux? Discuss its history in brief
2. 2.Explain the features of Linux Operating System in brief.

Shobhit
Institute of Engineering & Technology
Deemed to-be-University
EDUCATION EMPOWERS

Linux Distributions

Linux Distributions (Distros)

Other operating systems like Microsoft combine each bit of codes internally and release it as a single package. You have to choose from one of the version they offer.

But Linux is different from them. Different parts of Linux are developed by different organizations.

Different parts include kernel, shell utilities, X server, system environment, graphical programs, etc. If you want you can access the codes of all these parts and assemble them yourself. But its not an easy task seeking a lot of time and all the parts has to be assembled correctly in order to work properly.

Linux Distributions

Linux Distributions List

There are on an average six hundred Linux distributors providing different features. Here, we'll discuss about some of the popular Linux distros today.

1) Ubuntu

It came into existence in 2004 by Canonical and quickly became popular. Canonical wants Ubuntu to be used as easy graphical Linux desktop without the use of command line. It is the most well known Linux distribution. Ubuntu is a next version of Debian and easy to use for newbies. It comes with a lots of pre-installed apps and easy to use repositories libraries.

Earlier, Ubuntu uses GNOME2 desktop environment but now it has developed its own unity desktop environment. It releases every six months and currently working to expand to run on tablets and smartphones.

2) Linux Mint

Mint is based on Ubuntu and uses its repository software so some packages are common in both.

Linux Distributions

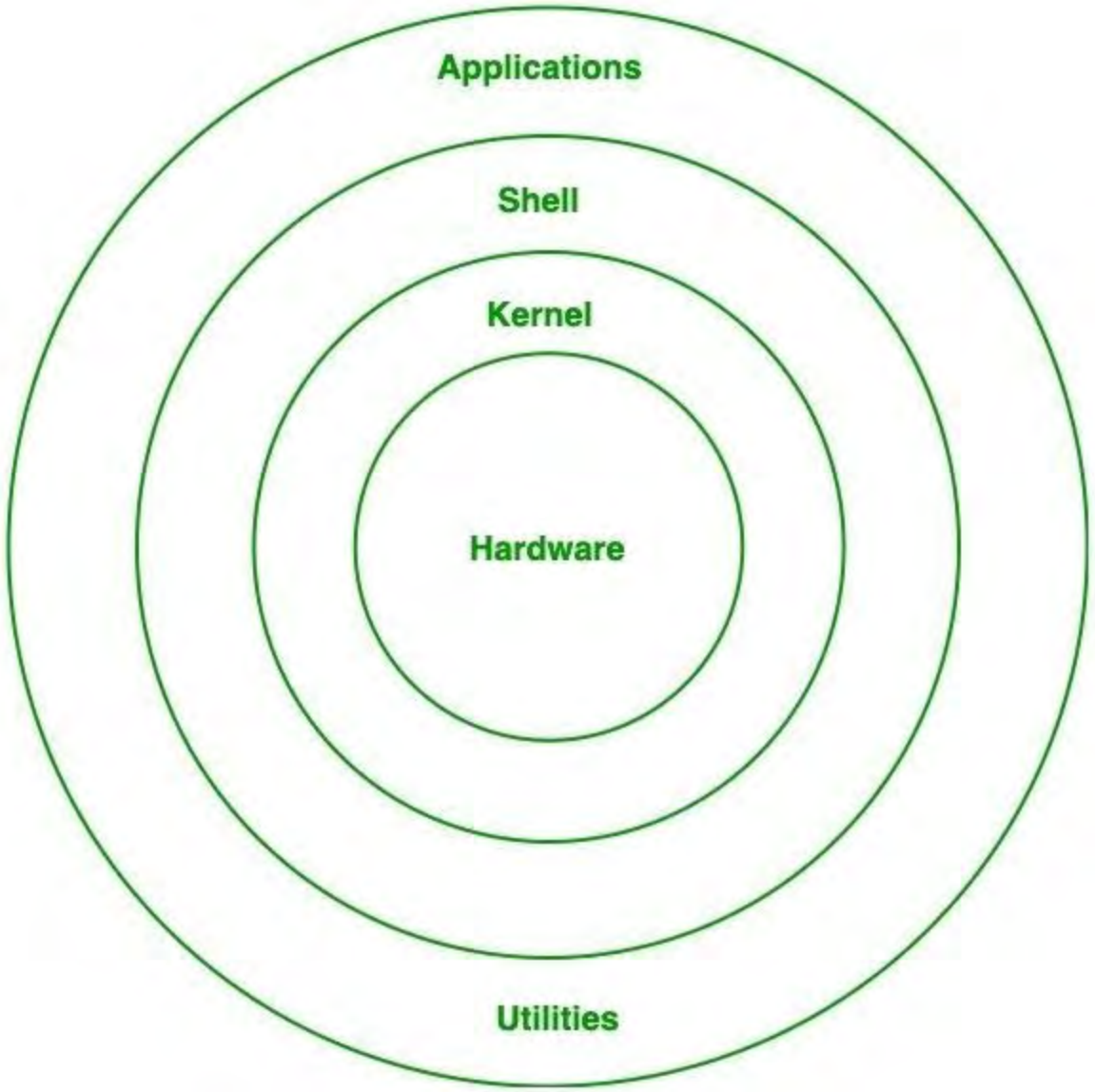
Earlier it was an alternative of Ubuntu because media codecs and proprietary software are included in mint but was absent in Ubuntu. But now it has its own popularity and it uses cinnamon and mate desktop instead of Ubuntu's unity desktop environment.

3) Debian

Debian has its existence since 1993 and releases its versions much slowly then Ubuntu and mint.

Distribution	Why To Use
UBuntu	It works like Mac OS and easy to use.
Linux mint	It works like windows and should be use by new comers.
Debian	It provides stability but not recommended to a new user.
Fedora	If you want to use red hat and latest software.
Red hat enterprise	To be used commercially.
CentOS	If you want to use red hat but without its trademark.
OpenSUSE	It works same as Fedora but slightly older and more stable.
Arch Linux	It is not for the beginners because every package has to be installed by yourself.

Linux O.S – Architecture



Linux O.S – Architecture

Kernel: Kernel is the core of the Linux based operating system. It virtualizes the common hardware resources of the computer to provide each process with its virtual resources. This makes the process seem as it is the sole process running on the machine. The kernel is also responsible for preventing and mitigating conflicts between different processes. Different types of the kernel are:

Monolithic Kernel

Hybrid kernels

Exo kernels

Micro kernels

System Library: It is the special types of functions that are used to implement the functionality of the operating system.

Shell: It is an interface to the kernel which hides the complexity of the **kernel's** functions from the users. It takes commands from the user and executes the **kernel's** functions.

Hardware Layer: This layer consists all peripheral devices like RAM/ HDD/ CPU etc.

System Utility: It provides the functionalities of an operating system to the user.

Linux O.S – Advantages and Disadvantages

Advantages of Linux

1. The main advantage of Linux, is it is an open-source operating system, means the source code is easily available for everyone and you are allowed to contribute, modify and distribute the code to anyone without any permissions.
2. In terms of security, Linux is more secure than any other operating system. It does not mean that Linux is 100 percent secure it has some malware for it but is less vulnerable than any other operating system. So, it does not require any anti-virus software.
3. The software updates in Linux are easy and frequent.
4. Various Linux distributions are available so that you can use them according to your requirements or according to your taste.
5. Linux is freely available to use on the internet.
6. It has large community support.
7. It provides high stability. It rarely slows down or freezes and there is no need to reboot it after a short time.

Linux O.S – Advantages and Disadvantages

8. It maintain the privacy of the user.
9. The performance of the Linux system is much higher than other operating systems. It allows a large number of people to work at the same time and it handles them efficiently.
10. It is network friendly.
11. The flexibility of Linux is high. There is no need to install a complete Linux suit you are allowed to install only required components.
12. Linux is compatible with a large number of file formats.
13. It is fast and easy to install from the web. It can also install in any hardware even in your old computer system.
14. It performs all tasks properly even if it has limited space on the hard disk.

Linux O.S – Advantages and Disadvantages

Disadvantages of Linux

1. It is not much user-friendly. So, it may be confusing for beginners.
2. It has small peripheral hardware drivers as compared to windows.
3. Is There Any Difference between Linux and Ubuntu?

The answer is YES. The main difference between Linux and Ubuntu is Linux is the family of open-source operating systems which is based on Linux kernel, whereas Ubuntu is a free open-source operating system and the Linux distribution which is based on Debian. Or in other words, Linux is the core system and Ubuntu is the distribution of Linux. Linux is developed by Linus Torvalds and released in 1991 and Ubuntu is developed by Canonical Ltd. and released in 2004.

ASSIGNMENT:

Ques:

1. Explain the Architecture Model of Linux O.S. with a neat diagram.
2. What are various Linux Distributions? Discuss them

Shobhit
Institute of Engineering & Technology
Deemed to-be-University
EDUCATION EMPOWERS

Linux O.S – Installation

Installing Linux

Let's look the various methods we can use to install Ubuntu.

Installing Linux using USB stick

This is one of the easiest methods of installing Ubuntu or any distribution on your computer. Follow the steps.

Step 1) Download the .iso or the OS files on your computer from this link.

Step 2) Download free software like 'Universal USB installer to make a bootable USB stick.

Step 3) Select an Ubuntu Distribution form the dropdown to put on your USB

Select your Ubuntu iso file download in step 1.

Select the drive letter of USB to install Ubuntu and Press create button.

Step 4) Click YES to Install Ubuntu in USB.

Step 5) After everything has been installed and configured, a small window will appear Congratulations! You now have Ubuntu on a USB stick, bootable and ready to go.

Step 3) Boot your computer from the optical drive and follow the instructions as they come.

Linux O.S – Installation

Installing Linux using CD-ROM

Those who like the way a CD runs should try using this method.

Step 1) Download the .iso or the OS files onto your computer from this link http://www.ubuntu.com/download/desktop.

Step 2) Burn the files to a CD.

Linux O.S – The BOOT Process

Press the power button on your system, and after few moments you see the Linux login prompt.

Have you ever wondered what happens behind the scenes from the time you press the power button until the Linux login prompt appears?

The following are the 6 high level stages of a typical Linux boot process.

1. BIOS

BIOS stands for Basic Input/Output System

Performs some system integrity checks

Searches, loads, and executes the boot loader program.

It looks for boot loader in floppy, cd-rom, or hard drive. You can press a key (typically F12 of F2, but it depends on your system) during the BIOS startup to change the boot sequence.

Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.

So, in simple terms BIOS loads and executes the MBR boot loader.

Linux O.S – The BOOT Process

2. MBR

MBR stands for Master Boot Record.

It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda

MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.

It contains information about GRUB (or LILO in old systems).

So, in simple terms MBR loads and executes the GRUB boot loader.

3. GRUB

GRUB stands for Grand Unified Bootloader.

If you have multiple kernel images installed on your system, you can choose which one to be executed.

GRUB displays a splash screen, waits for few seconds, if you **don't** enter anything, it loads the default kernel image as specified in the grub configuration file.

GRUB has the knowledge of the file system (the older Linux loader LILO **didn't** understand file system).

Linux O.S – The BOOT Process

Grub configuration file is /boot/grub/grub.conf (/etc/grub.conf is a link to this).

The following is sample grub.conf of CentOS.

```
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
hiddenmenu
title CentOS (2.6.18-194.el5PAE)
        root (hd0,0)
        kernel /boot/vmlinuz-2.6.18-194.el5PAE ro root=LABEL=/
        initrd /boot/initrd-2.6.18-194.el5PAE.img
```

As you notice from the above info, it contains kernel and initrd image.

So, in simple terms GRUB just loads and executes Kernel and initrd images.

Linux O.S – The BOOT Process

4. Kernel

Mounts the root file system as specified in the **"root="** in grub.conf

Kernel executes the /sbin/init program

Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a **'**ps -ef | grep **init'** and check the pid.

initrd stands for Initial RAM Disk.

initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

5. Init

Looks at the /etc/inittab file to decide the Linux run level.

Following are the available run levels

0 – halt

1 – Single user mode

2 – Multiuser, without NFS

3 – Full multiuser mode

4 – unused

5 – X11

Linux O.S – The BOOT Process

6 – reboot

Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.

Execute 'grep initdefault /etc/inittab' on your system to identify the default run level

If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.

Typically you would set the default run level to either 3 or 5.

6. Runlevel programs

When the Linux system is booting up, you might see various services getting started. For example, it might say **"starting** sendmail **…. OK"**. Those are the runlevel programs, executed from the run level directory as defined by your run level.

Depending on your default init level setting, the system will execute the programs from one of the following directories.

Run level 0 – /etc/rc.d/rc0.d/

Run level 1 – /etc/rc.d/rc1.d/

Run level 2 – /etc/rc.d/rc2.d/

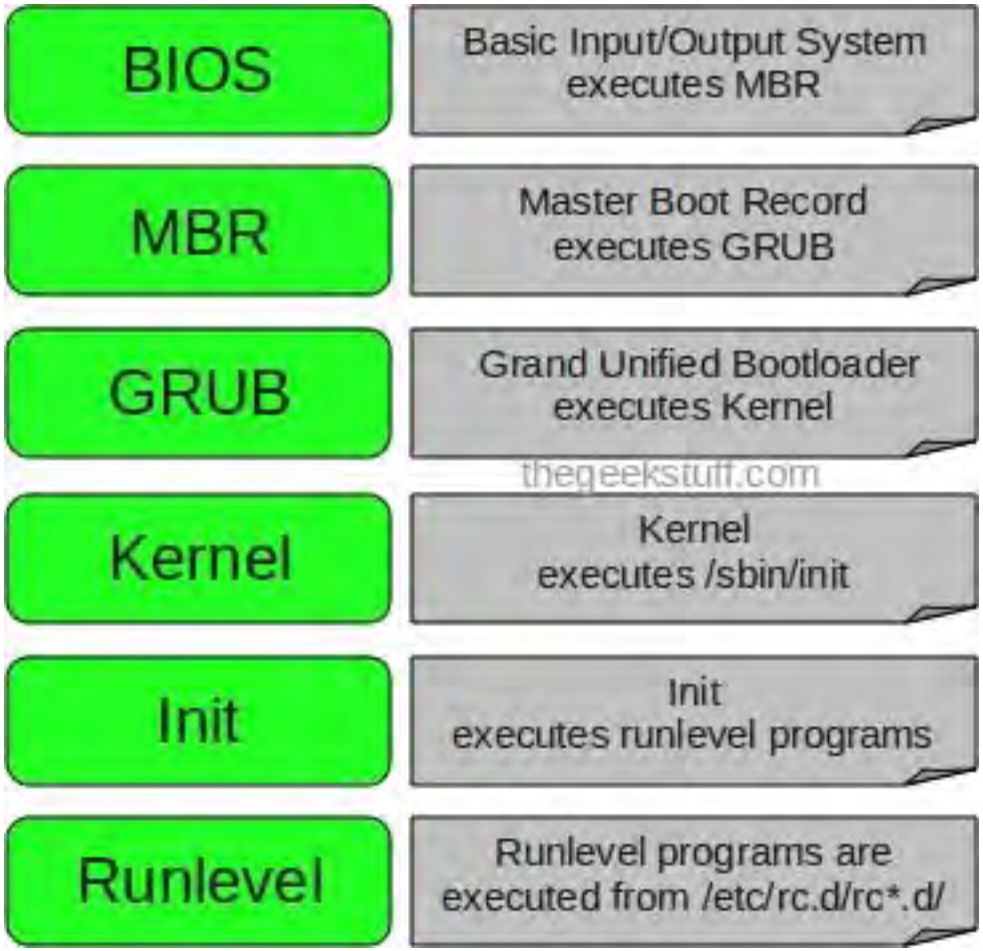
Run level 3 – /etc/rc.d/rc3.d/

Run level 4 – /etc/rc.d/rc4.d/

Run level 5 – /etc/rc.d/rc5.d/

Run level 6 – /etc/rc.d/rc6.d/

Linux O.S – The BOOT Process



Linux O.S – LOGIN Process

The Login Process

The following step-by-step description shows what happens each time a user logs in to a UNIX / Linux computer system.

Users enters their username.

User enters their password.

The operating system confirms your name and password.

A "shell" is created for you based on your entry in the "/etc/passwd" file (in small businesses, this is usually a Bourne Shell).

You are "placed" in your "home"directory.

Startup information is read from the file named "/etc/profile". This file is known as the system login file. When every user logs in, they read the information in this file.

Additional information is read from the file named ".profile" that is located in your "home" directory. This file is known as your personal login file. (This is the file that usually contains the "menu" program.)

Linux O.S – Shutdown Process

The shutdown command brings the system down in a secure way. All logged-in users are notified that the system is going down, and login operations are blocked. It is possible to shut the system down immediately, or after a specified delay.

All processes are first notified that the system is going down by the signal SIGTERM. This gives programs like vi time to save the file being edited, mail and news processing programs a chance to exit cleanly, etc.

shutdown does its job by signalling the init process, asking it to change the runlevel. Runlevel 0 is used to halt the system, runlevel 6 is used to reboot the system, and runlevel 1 is used to put the system into a state where administrative tasks can be performed (single-user mode). Runlevel 1 is the default, unless the -h or -r options are specified.

Syntax

shutdown [-akrhPHfFnc] [-t sec] time [message]

Linux O.S – Shutdown Process

Options

-a	Control access to the **shutdown** command using the control access file **/etc/shutdown.allow**. See Access Control below for more information.
-k	Do not shut down, but send the warning messages as if the shutdown were real.
-r	Reboot after shutdown.
-h	Instructs the system to shut down and then halt.
-P	Instructs the system to shut down and then power down.
-H	If **-h** is also specified, this option instructs the system to drop into boot monitor on systems that support it.
-f	Skip **fsck** after reboot.
-F	Force **fsck** after reboot.
-n	Don't call **init** to do the shutdown of processes; instruct **shutdown** to do that itself. The use of this option is discouraged, and its results are not always predictable.
-c	Cancel a pending shutdown. (This does not apply to "**shutdown now**", which does not wait before shutting down.) With this option, it is not possible to give the *time* argument, but you can still specify an explanatory message that will be sent to all users.
-t *sec*	Tell **init** to wait *sec* seconds between sending processes the warning and the kill signal, before changing to another runlevel.
time	The *time* argument specifies when to perform the shutdown operation.

Linux O.S – Shutdown Process

- If a shutdown is scheduled for the future, it will create the advisory file /etc/nologin which causes programs such as login not to allow new user logins. This file is created five minutes before the shutdown sequence starts. shutdown removes this file if it is stopped before it can signal init (i.e. it is cancelled or something goes wrong). It also removes it before calling init to change the runlevel.

- The -f flag means "reboot fast". This only creates an advisory file /fastboot which can be tested by the system when it comes up again. The system boot rc file ("rc" stands for "runcom", which is short for "run commands") can test if this file is present, and decide not to run fsck since the system has been shut down in the proper way. After that, the boot process should remove /fastboot.

ASSIGNMENT:

Ques:

1. How you will install Linux OS using a Pen drive? Explain
2. Discuss the shutdown process of Linux OS in brief.

Shobhit
Institute of Engineering & Technology
Deemed to-be-University
EDUCATION EMPOWERS

Linux O.S – User Management

User management includes everything from creating a user to deleting a user on your system. User management can be done in three ways on a Linux system.

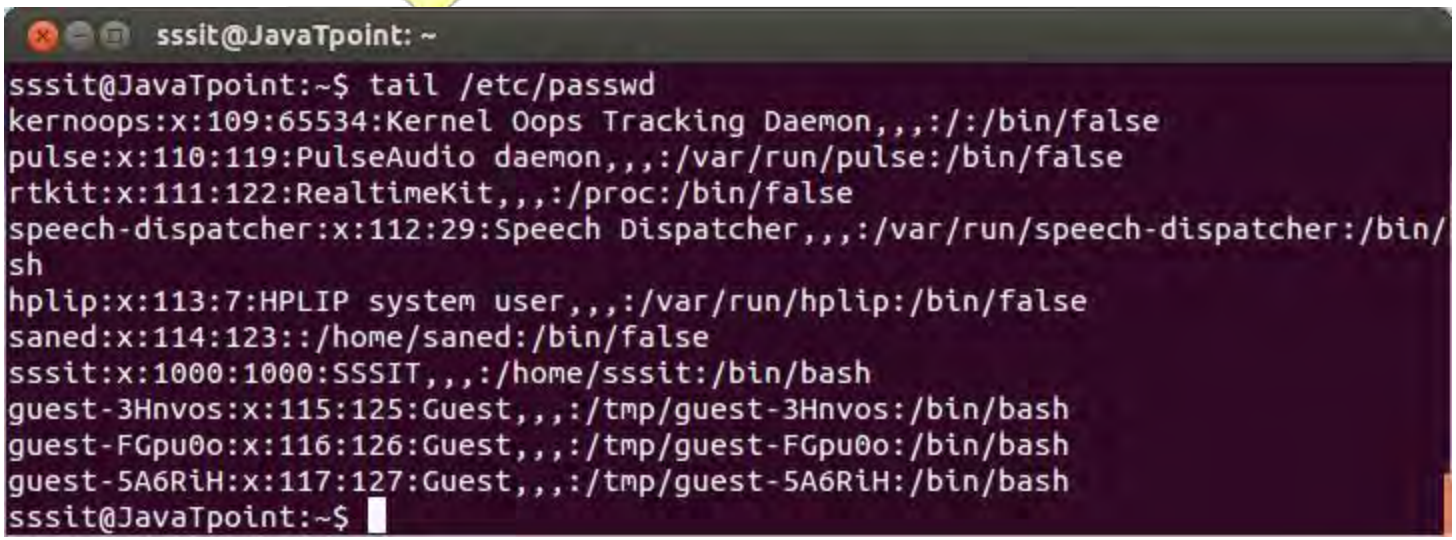
Graphical tools are easy and suitable for new users, as it makes sure you'll not run into any trouble.

Command line tools includes commands like useradd, userdel, passwd, etc. These are mostly used by the server administrators.

Third and very rare tool is to edit the local configuration files directly using vi.

/etc/passwd

The local user database in Linux is /etc/passwd directory.

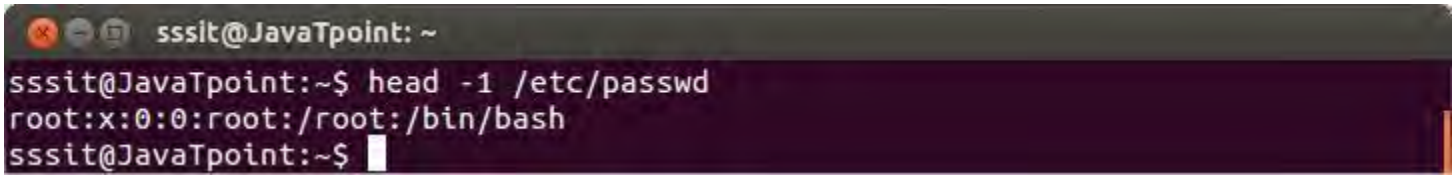


Linux O.S – User Management

Look at the above snapshot, it has seven columns separated by a colon. Starting from the left columns denotes username, an x, user id, primary group id, a description, name of home directory and a login shell.

root

The root user is the superuser and have all the powers for creating a user, deleting a user and can even login with the other user's account. The root user always has userid 0.



Linux O.S – User Management

useradd

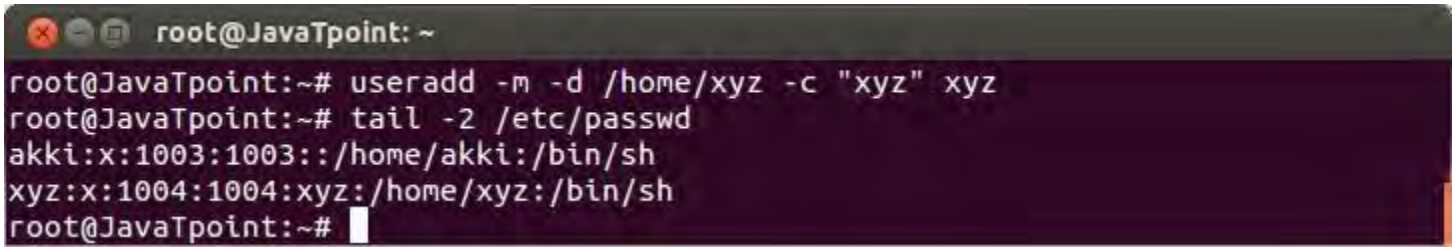
With useradd commands you can add a user.

Syntax:

useradd -m -d /home/<userName> -c "<userName>" <userName>

Example:

useradd -m -d /home/xyz -c "xyz" xyz



Linux O.S – User Management

userdel

To delete a user account userdel command is used.

Syntax:

userdel -r <userName>

Linux Local User Management5

Example:

userdel -r xyz

Look at the below snapshot, first we have shown the xyz user account with 'tail' command. To delete it, command "userdel -r xyz" is passed.

To recheck, again 'tail' command is passed and as you can see no xyz user account is displayed.

Hence, it is deleted.

Linux O.S – User Management

usermod

The command usermod is used to modify the properties of an existing user.

Syntax:

usermod -c <'newName'> <oldName>

Example:

usermod -c 'jhonny' john

Deleting Home Directories

By using userdel -r option, you can delete home directory along with user account.

Syntax:

userdel -r <userName>

Example:

userdel -r john

Linux O.S – User Management

Login Shell

The /etc/passwd file also tells about the login shell for the user.

Linux Local User Management9

Look at the above snapshot, user guest will log in with /bin/bash shell and user jtp will log in with /bin/ksh shell.

You can change the shell mode with usermod command for a user.

Syntax:

usermod -s <newShell> <userName>

Example:

usermod -s /bin/bash jtp

Linux O.S – User Management

chsh

Users can change their login shell with chsh command.

Both the command chsh and chsh -s will work to change the shell.

Syntax:

chsh

Linux O.S – User Groups

Linux Groups

Users can be listed in different groups. Group allow us to set permission on the group level instead of setting the permission on individual level.

Every Linux distribution have a graphical tool to manage groups. Groups can be managed by graphical tools, command line tools and by vi or vigr depending upon the user's experience. Only experienced users should use vi or vigr to manage groups, since it will do proper locking or changes in the file.

groupadd

The groupadd command creates or add a group in our system.

Syntax:

groupadd <groupName>

Example:

groupadd php

groupadd java

groupadd android

groupadd spring

Linux Local Group1

Look at the above snapshot, groups php, java, android and spring are created with groupadd command.

Linux O.S – User Groups

Group File

The /etc/group file defines the group membership. A user can be a member of more than one group.

Syntax:

/etc/group

Linux Local Group2

Look at the above snapshot, first column indicates group name, second is the group's encrypted password which may remain empty also, third is group identification (GID) and fourth is the list of members. Fourth list is empty as these groups do not have members.

Groups

The group command tells about the group where current user belongs to.

Syntax:

groups

Linux Local Group3

Look at the above snapshot, user jtp and sssit belongs to the different groups.

Linux O.S – User Groups

usermod

The group members can be edited with usermod or useradd command. If a group is not listed then by default, usermod command will remove the user from every group of which he is a member. Here, -a (append) option is used to prevent this from happening.

Syntax:

usermod -a -G <group> <userName>

Example:

usermod -a -G php akki

usermod -a -G php abc

usermod -a -G java jtp

Linux Local Group4

Look at the above snapshot, we have displayed the list of /etc/group. User akki and abc are added into the group php, user jtp is added into java.

groupmod

With the help of groupmod command you can change the name of an already existing group.

Syntax:

groupmod -n <oldGroup> <newGroup>

Example:

groupmod -n sql spring

Linux O.S – User Groups

gruopdel

The command groupdel will delete a group permanently from the system.

Syntax:

groupdel <group>

Example:

groupdel sql

gpasswd

Control of group membership can be passed on to another user with gpasswd command.

Syntax:

gpsswd -A <user> <group>

Example:

gpasswd -A jtp java

Linux O.S – User Groups

gruopdel

The command groupdel will delete a group permanently from the system.

Syntax:

groupdel <group>

Example:

groupdel sql

gpasswd

Control of group membership can be passed on to another user with gpasswd command.

Syntax:

gpsswd -A <user> <group>

Example:

gpasswd -A jtp java

Linux O.S – User Groups

Group administrators need not to be a member of the group. They can add or remove a member without being a member of that group.

File /etc/gshadow keeps the information about the group administrators as shown in below snapshot.

Syntax:

gpasswd -A "" <group>

Example:

gpasswd -A "" java

Linux O.S – User Groups

Group administrators need not to be a member of the group. They can add or remove a member without being a member of that group.

File /etc/gshadow keeps the information about the group administrators as shown in below snapshot.

Syntax:

gpasswd -A "" <group>

Example:

gpasswd -A "" java

ASSIGNMENT:

Ques:

1. What are the powers of root user? Discuss.
2. What are various user groups used in Linux? Discuss with the example.

Shobhit
Institute of Engineering & Technology
Deemed to-be-University
EDUCATION EMPOWERS

Linux O.S – Shell Script and Commands

A shell script is a computer program designed to be run by the Unix/Linux shell which could be one of the following:

The Bourne Shell

The C Shell

The Korn Shell

The GNU Bourne-Again Shell

A shell is a command-line interpreter and typical operations performed by shell scripts include file manipulation, program execution, and printing text.

Types of Shell

There are two main shells in Linux:

1. The Bourne Shell: The prompt for this shell is $ and its derivatives are listed below:

- POSIX shell also is known as sh
- Korn Shell also knew as sh
- Bourne Again Shell also knew as bash (most popular)

2. The C shell: The prompt for this shell is %, and its subcategories are:

C shell also is known as csh. Tops C shell also is known as tcsh

Linux O.S – Shell Script and Commands

What Is Shell Scripting?

SHELL SCRIPTING is writing a series of commands for the shell to execute. It can combine lengthy and repetitive sequences of commands into a single and simple script, which can be stored and executed anytime. This reduces the effort required by the end user.

Let us understand the steps in creating a Shell Script

Create a file using a vi editor(or any other editor). Name script file with extension .sh
Start the script with #! /bin/sh
Write some code.
Save the script file as filename.sh
For executing the script type bash filename.sh
"#!" is an operator called shebang which directs the script to the interpreter location. So, if we use"#! /bin/sh" the script gets directed to the bourne-shell.

Let's create a small script -

Linux O.S – Shell Script and Commands

What are Shell Variables?

As discussed earlier, Variables store data in the form of characters and numbers. Similarly, Shell variables are used to store information and they can by the shell only.

For example, the following creates a shell variable and then prints it:

```
variable ="Hello"
echo $variable
```

Below is a small script which will use a variable.


```
#!/bin/sh
echo "what is your name?"
read name
echo "How do you do, $name?"
read remark
echo "I am $remark too!"
```

Linux O.S – Shell Script and Commands

What are Shell Variables?

As discussed earlier, Variables store data in the form of characters and numbers. Similarly, Shell variables are used to store information and they can by the shell only.

For example, the following creates a shell variable and then prints it:

```
variable ="Hello"
echo $variable
```

Below is a small script which will use a variable.


```
#!/bin/sh
echo "what is your name?"
read name
echo "How do you do, $name?"
read remark
echo "I am $remark too!"
```

ASSIGNMENT:

Ques:

1. Discuss some shell scripts in detail.
2. What are various shells in Linux Programming

Shobhit
Institute of Engineering & Technology
Deemed to-be-University
EDUCATION EMPOWERS

The grep family

What is grep?

- A text manipulation program
- Used to find pattern in files or text
- global regular expression print (: g/RE/p) / general regular expression parser (grep)
- Other text manipulation commands – cut, tr, awk, sed
- grep family - grep, egrep, and fgrep
- Type **man grep** to find list of options

The grep command syntax:

General syntax of grep command

```
grep [-options] pattern [filename]
```

Examples:

```
$ grep pattern filename
$ grep pattern file1 file2
$ grep -i desktop /etc/services
$ grep [yf] /etc/group
$ grep -vi tcp /etc/services
$ ip addr show | grep inet
```

Examples: grep options

```
$ grep -i desktop /etc/services
$ grep -vi tcp /etc/services
$ grep -v apple fruitlist.txt
$ grep -l 'main' *.c
```

lists the names of all C files in the current directory whose contents mention 'main'.

```
$ grep -r 'hello' /home/gigi
```

searches for 'hello' in all files under the '/home/gigi' directory

```
$ grep -w 'north' datafile
```

Only the line containing the word north is printed, not northwest

```
$ grep -c "Error" logfile.txt
```

EXAMPLE

Option	Description
-b	Display the block number at the beginning of each line.
-c	Display the number of matched lines.
-h	Display the matched lines, but do not display the filenames.
-i	Ignore case sensitivity.
-l	Display the filenames, but do not display the matched lines.
-n	Display the matched lines and their line numbers.
-s	Silent mode.
-v	Display all lines that do NOT match.
-w	Match whole word.

What is sed?

- **sed** is a *non-interactive* stream-oriented UNIX utility
- **sed** is used for parsing text files, and to apply textual transformations to a sequential stream of data
- **sed** reads the input data stream line by line, applies the operations that have been specified, and then outputs the modified data
  - `$ sed 's/needle/magnet/g' haystack > new_haystack`
- **sed** is often used as a *filter* in a pipeline
- **sed** has its origins in **ed**, the original UNIX line editor
- Basic difference between **sed** and **ed** is that **ed** is not stream oriented, whereas **sed** is
- **ed** is an interactive editor, whereas **sed** is not

What is awk?

AWK can be described as: A Pattern-Matching Programming Language

AWK is designed for processing text-based data, either in files or data streams

A typical example of an awk program is one that transforms data into a formatted report

You can trace the lineage of **awk** to **sed** and **grep**, and through these two programs to **ed**, the original UNIX line editor

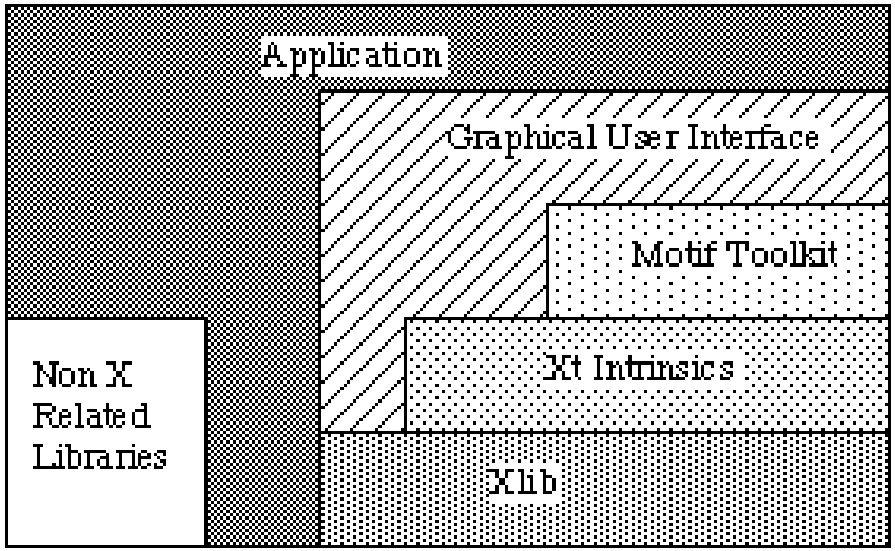
X-Programming Model

- The client and server are connected by a communication path called (surprise, surprise) the connector . This is performed by a low-level C language interface known as Xlib . Xlib is the lowest level of the X system software hierarchy or architecture (Fig 3.1). Many applications can be written using Xlib alone. However, in general, it will be difficult and time consuming to write complex GUI programs only in Xlib. Many higher level subroutine libraries, called toolkits , have been developed to remedy this problem.

- Note: X is not restricted to a single language, operating system or user interface. It is relatively straightforward to link calls to X from most programming languages. An X application must only be able to generate and receive messages in a special form, called X protocol messages . However, the protocol messages are easily accessible as C libraries in Xlib (and others).

- There are usually two levels of toolkits above Xlib

- X Toolkit (Xt) Intrinsics are parts of the toolkit that allow programmers to build new widgets.
- Third Party Toolkits -- such a Motif .
- An application program in X will usually consist of two parts. The graphical user interface written in one or more of Xlib, Xt or Motif and the algorithmic or functional part of the application where the input from the interface and other processing tasks are defined. Fig. 3.1 illustrates the relationships between the application program and the various parts of the X System.



The main concern of this text is to introduce concepts in building the graphical user interface in X and Motif in particular. We now briefly describe the main tasks of the three levels of the X programming model before embarking on writing Motif programs.

X Lib and Xt Intrinsics

•The main task of Xlib is to translate C data structures and procedures into the special form of X protocol messages which are then sent off. Obviously the converse of receiving messages and converting them to C structures is performed as well. Xlib handles the interface between client (application) and the network.

•Toolkits implement a set of user interface features or application environments such as menus, buttons or scroll bars (referred to as widgets).
•They allow applications to manipulate these features using object-oriented techniques.

•X Toolkit Intrinsics or Xt Intrinsics are a toolkit that allow programmers to create and use new widgets.

•If we use widgets properly, it will simplify the X programming process and also help preserve the look and feel of the application which should make it easier to use.

•We will have to call some Xt functions when writing Motif programs since Motif is built upon Xt and thus needs to use Xt. However, we do notneed to fully understand the workings of Xt as Motif takes care of most things for us.

Widget Classes and Hierarchies

A ***widget***, in Motif, may be regarded as a general abstraction for user-interface components. Motif provides widgets for almost every common GUI component, including buttons, menus and scroll bars. Motif also provides widgets whose only function is to control the layout of other widgets -- thus enabling fairly advanced GUIs to be easily designed and assembled.

A widget is designed to operate independently of the application except through well defined interactions, called ***callback functions*** . This takes a lot of mundane GUI control and maintenance away from the application programmer. Widgets know how to redraw and highlight themselves, how to respond to certain events such as a mouse click ***etc.*** Some widgets go further than this, for example the Text widget is a fully functional text editor that has built in cut and paste as well as other common text editing facilities.

The general behaviour of each widget is defined as part of the Motif (Xm) library. In fact Xt defines certain base classes of widgets which form a common foundation for nearly all Xt based widget sets. Motif provides a ***widget set***, the Xm library, which defines a complete set of widget classes for most GUI requirements on top of Xt (Fig 3.1).

The Motif Reference Manual provides definitions on all aspects of widget behaviour and interaction. Basically, each widget is defined as a C data structure whose elements define a widget's data attributes, or ***resources*** and pointers to functions, such as ***callbacks***.

Each widget is defined to be of a certain ***class*** . All widgets of that class inherit the same set of resources and callback functions. Motif also defines a whole hierarchy of widget classes. There are two broad Motif widget classes that concern us.

The ***Primitive*** widget class contains actual GUI components, such as buttons and text widgets. The ***Manager*** widget class defines widgets that hold other widgets.

Chapter 4 introduces basic Motif widget programming concepts and introduces how resources and callback functions are set up. Chapter 5 then goes on to fully define each widget class and the Motif widget class hierarchy. Following Chapters then address each widget class in detail.

Introduction

- The X Window System (X11) is an open source, cross platform, client-server computer software system that provides a GUI in a distributed network environment.

- Used primarily on Unix variants, X versions are also available for other operating systems. Features of the X window system include network transparency, the ability to link to different networks, and customizable graphical capabilities. The X window system was first developed in 1984, as part of project Athena, a collaboration between Stanford University and MIT. X.Org Foundation, an open group, manages the development and standardization of the X window system.

- The X Window System is also known simply as X, X11 or X Windows.

WORKING

The client/server model in X system works in reverse to typical client/server model, where the client runs on the local machine and asks for services from the server. In X system, the server runs on the local machine and provides its display and services to the client programs. The client programs may be local or remotely exist over different networks, but appear transparently.

X is used in networks of interconnected mainframes, minicomputers, workstations, and X Terminals. X window system consists of a number of interacting components, including:

X server:

Manages the display and input hardware. It captures command-based and graphics-based inputs from input hardware and passes it to the client application that requested it. It also receives inputs from the client applications and displays the output under guidance from windows manager.

The only component that interacts with hardware is X server. This makes it easier to recode it as per the requirements of different hardware architectures.

Windows Manager:

Is the client application that manages client windows. It controls the general operations of the window system like geometry, appearance, coordinates, and graphical properties of X display. Window manager can change the size and position of windows on the display and reshuffle windows in a window stack.

X client:

Is an application program that communicates with X server using X protocol. Xterm, Xclock, and Xcalc are examples of X clients. X manages its windows in a hierarchal structure. The shaded area that fills the entire screen is the root window. X client application windows are displayed on top of the root window and are often called the children of the root.

Event Handling

➢An event is data generated asynchronously by the X server as a result of some device activity or as side effects of a request sent by an Xlib function. Device-related events propagate from the source window to ancestor windows until some client application has selected that event type or until the event is explicitly discarded. The X server generally sends an event to a client application only if the client has specifically asked to be informed of that event type, typically by setting the event-mask attribute of the window. The mask can also be set when you create a window or by changing the window's event-mask. You can also mask out events that would propagate to ancestor windows by manipulating the do-not-propagate mask of the window's attributes. However, MappingNotify events are always sent to all clients.

Event Types

- An event type describes a specific event generated by the X server. For each event type, a corresponding constant name is defined in X11/X.h, which is used when referring to an event type. The following table lists the event category and its associated event type or types. The processing associated with these events is discussed in section "Event Processing Overview".

- Event Category — Event Type
- Keyboard events — KeyPress, KeyRelease
- Pointer events — ButtonPress, ButtonRelease, MotionNotify
- Window crossing events — EnterNotify, LeaveNotify
- Input focus events — FocusIn, FocusOut
- Keymap state notification event — KeymapNotify
- Exposure events — Expose, GraphicsExpose, NoExpose
- Structure control events — CirculateRequest, ConfigureRequest, MapRequest, ResizeRequest
- Window state notification events — CirculateNotify, ConfigureNotify, CreateNotify, DestroyNotify, GravityNotify, MapNotify, MappingNotify, ReparentNotify, UnmapNotify, VisibilityNotify
- Colormap state notification event — ColormapNotify
- Client communication events — ClientMessage, PropertyNotify, SelectionClear, SelectionNotify, SelectionRequest

Event Structure

For each event type, a corresponding structure is declared in **X11/Xlib.h**. All the event structures have the following common members:
typedef struct { int type; unsigned long serial; /* # of last request processed by server */ Bool send_event; /* true if this came from a SendEvent request */ Display *display; /* Display the event was read from */ Window window; } XAnyEvent; The **type** member is set to the event type constant name that uniquely identifies it. For example, when the X server reports a GraphicsExpose event to a client application, it sends an XGraphicsExposeEvent structure with the **type** member set to GraphicsExpose. The **display** member is set to a pointer to the display the event was read on. The **send_event** member is set to **True** if the event came from a SendEvent protocol request. The **serial** member is set from the serial number reported in the protocol but expanded from the 16-bit least-significant bits to a full 32-bit value. The **window** member is set to the window that is most useful to toolkit dispatchers.
The X server can send events at any time in the input stream. Xlib stores any events received while waiting for a reply in an event queue for later use. Xlib also provides functions that allow you to check events in the event queue (see section "Event Queue Management").

In addition to the individual structures declared for each event type, the **XEvent** structure is a union of the individual structures declared for each event type. Depending on the type, you should access members of each event by using the **XEvent** union.
typedef union _XEvent {

Event Structure

```
typedef union _XEvent {
        int type;       /* must not be changed */
        XAnyEvent xany;
        XKeyEvent xkey;
        XButtonEvent xbutton;
        XMotionEvent xmotion;
        XCrossingEvent xcrossing;
        XFocusChangeEvent xfocus;
        XExposeEvent xexpose;
        XGraphicsExposeEvent xgraphicsexpose;
        XNoExposeEvent xnoexpose;
        XVisibilityEvent xvisibility;
        XCreateWindowEvent xcreatewindow;
        XDestroyWindowEvent xdestroywindow;
        XUnmapEvent xunmap;
        XMapEvent xmap;
        XMapRequestEvent xmaprequest;
        XReparentEvent xreparent;
        XConfigureEvent xconfigure;
        XGravityEvent xgravity;
        XResizeRequestEvent xresizerequest;
        XConfigureRequestEvent xconfigurerequest;
        XCirculateEvent xcirculate;
        XCirculateRequestEvent xcirculaterequest;
        XPropertyEvent xproperty;
        XSelectionClearEvent xselectionclear;
        XSelectionRequestEvent xselectionrequest;
        XSelectionEvent xselection;
        XColormapEvent xcolormap;
        XClientMessageEvent xclient;
        XMappingEvent xmapping;
        XErrorEvent xerror;
        XKeymapEvent xkeymap;
        long pad[24];
} XEvent;
```

An XEvent structure's first entry always is the type member, which is set to the event type. The second member always is the serial number of the protocol request that generated the event. The third member always is send_event, which is a Bool that indicates if the event was sent by a different client. The fourth member always is a display, which is the display that the event was read from. Except for keymap events, the fifth member always is a window, which has been carefully selected to be useful to toolkit dispatchers. To avoid breaking toolkits, the order of these first five entries is not to change. Most events also contain a time member, which is the time at which an event occurred. In addition, a pointer to the generic event must be cast before it is used to access any other information in the structure.