CHAPTER 1

Introduction to Unix

1.1 INTRODUCTION TO OPERATING SYSTEM

In the annals of computer science, the most commendable development one could consider is the emergence of the operating system which enables even a lay man to avail the services of computers without joining computer science program! An Operating System is the software layer between the hardware and user (shown in Figure 1.1) giving a compact and convenient interface to the user.

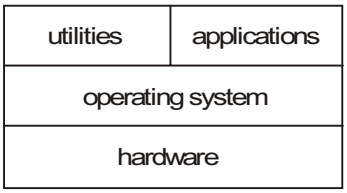

Figure 1.1 A Typical Operating System.

An Operating System is mainly responsible for

- Device management using device drivers
- Process management using processes and threads
- Inter-process communication
- Memory management
- File systems

Also, all OS's come with a set of standard utilities for performing tasks such as

- being able to start and stop processes
- being able to organize the set of available applications
- organize files into sets such as directories
- view files and sets of files
- edit files
- rename, copy, delete files
- communicate between processes

- sending messages through email
- downloading files from other machines

The Kernel

The *kernel* of an operating system is the part responsible for all other operations. When a computer boots up, it goes through some initialization activities, such as checking memory. It then loads the kernel and switches control to it. The kernel then starts up all the processes needed to communicate with the users and the rest of the environment (e.g. the LAN). The kernel is always loaded into memory, and the kernel functions run continuously, handling processes, memory, files and devices. The traditional structure of a kernel is a *layered* system, as in UNIX where all layers are part of the kernel, and each layer can talk to only a few other layers. Application programs and utilities lie above the kernel.

The UNIX kernel looks like (Figure 1.2)

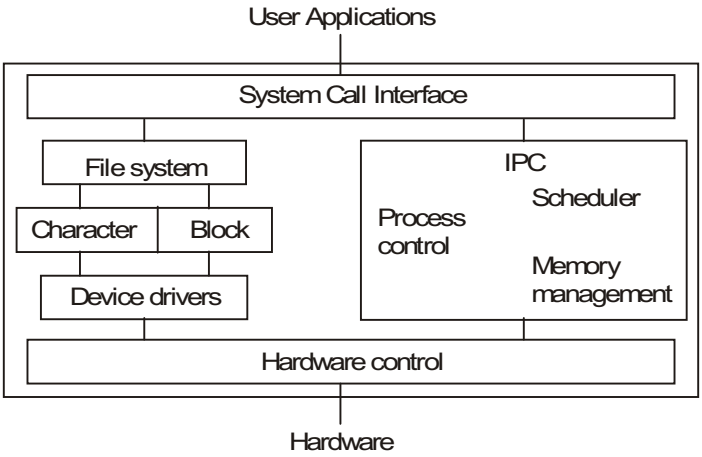

Figure 1.2 UNIX Kernel.

Most of the currently prevalent Operating Systems use instead a *micro kernel*, of minimal size. Many traditional kernel operations are made into user level services. Communication bearing service is often carried out by an explicit *message passing* mechanism. Mach is one of the major micro-kernel operating systems whose concepts are used by many others (see Figure 1.3).

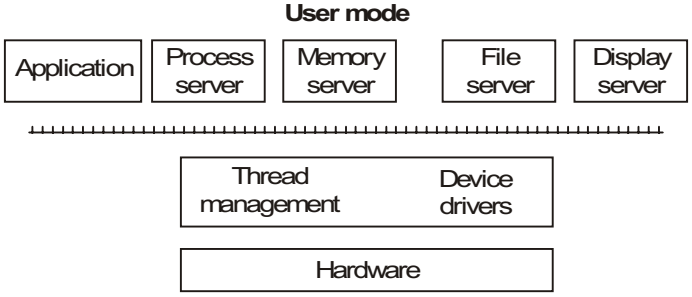

Figure 1.3 Micro Kernel Architecture.

Some systems, such as Windows NT (Figure 1.4) use a mixed approach [Gary Nutt].

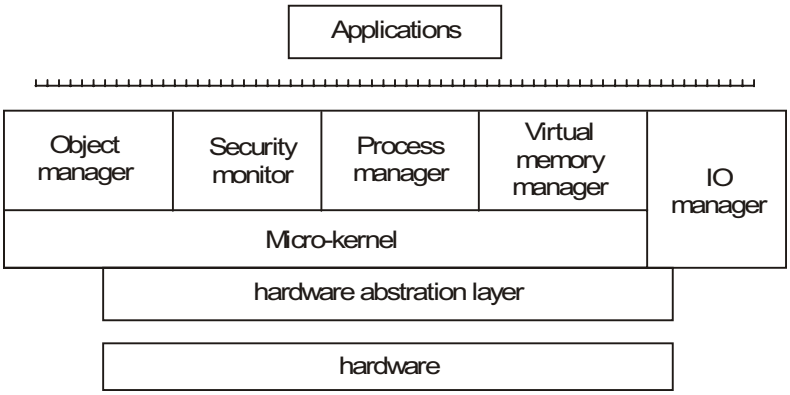

Figure 1.4 Windows NT Kernel.

The Distinguished Applications

An Operating System has been described as an "application with no top" [Venkateswarlu]. *Other* applications interact with it, through a large variety of entry points. In order to use an O/S, you need to be supplied with at least some applications that already use these entry points.

All Operating Systems come bundled with a set of "utilities" which do this. For example

- Windows95 has a shell that allows programs to be started from the Start button. There is a standard set of applications supplied
- MSDOS starts up with COMMAND.COM to supply a command line prompt, and a set of utilities
- UNIX has a set of command line shells and a huge variety of command line utilities
- X-Windows supplies a login shell (xdm). Others supply file managers, session managers, etc which can be used to provide a variety of interfaces to the underlying UNIX/POSIX system.

The Command Interpreter

Users interact with operating systems through the intermediary of a command interpreter. This utility responds to user inputs in the following ways:

- start or stop applications
- allow the user to switch control between applications
- allow control over communication between an application and other applications or the user.

The command interpreter may be character-based, as in the UNIX shells or MSDOS COMMAND.COM, or can be a GUI shell like the Windows 3.1 Program Manager. The interpreter may be simple, or with the power of a full programming language. It may be imperative (as in the UNIX shells), use message passing (as in AppleScript) or use visual programming metaphors such as drag-and-drop for object embedding (as in Microsoft's OLE). It is important to distinguish between the command interpreter and the underlying Operating System. The command interpreter may only use a subset of the capabilities offered by the Operating System; it may offer them in a clumsy or sophisticated way; it may require complex skills or be intended for novices

Differences between DOS and UNIX

- UNIX is multi user and multi tasking operating system whereas DOS is single user, single task system.
- All the commands in UNIX should be given in lower case while the DOS commands are case insensitive.
- Unlike UNIX, DOS is more virus prone.
- Processor will be in protected mode in UNIX whereas DOS uses unprotected mode.
- DOS uses only 640KB of RAM during boot time unlike UNIX which uses all the available RAM.
- UNIX needs an administrator which is not the case with DOS.
- UNIX employs time sharing operating system. Where as DOS supports a pseudo time sharing known as Terminate and Stay Resident (TSR) programs.
- UNIX supports both character user interface and graphical interface (X Windows) unlike DOS which supports only character user interface.
- User requires legal username and password to use UNIX machines. DOS systems can be used by any one without any username and password.
- UNIX uses single directory tree (/) irrespective of how many drives or partitions are there. Where as in DOS, a separate directory tree exists for each partition.
- UNIX supports NFS to share files.
- Till recently, DOS did not have proper WWW browser.

Linux and the Open Source Movement

Linus Torvalds [Matt Welsh]**,** a student at the University of Helsinki, created the first version of "Linux" in August 1991. Released as an open-source software under the Free Software Foundation's GNU General Public License (GPL), Linux quickly grew into a complete operating-system package, with contributions from hundreds of programmers. Since the release of version 1.0 in 1994, organizations have been able to download free copies of Linux. One could also purchase commercial distributions of Linux from companies such as **Slackware**, **Red Hat** etc who also provide consultancy, services, and maintenance [Carla Schroder].

Many people rise a question about Linux "if it's released under the Free Software Foundation's GPL, shouldn't it be free?". The answer is **no**. A company can charge money for products that include Linux, as long as the source code is made available. The GPL allows people to distribute (and charge for) their own versions of free software [Carla Schroder]. According to the Free Software Foundation, the "free" in free software refers to *freedom* or ***liberty***, ***not price***. In the foundation's definition, organizations have the freedom to run software for any purpose, study how it works, modify, improve and re-release it.

Another common misconception about Linux is that it's a complete operating system. In reality Linux refers to the " kernel or core " of the operating system. Combining Linux with a set of open-source GNU programs from the Free Software Foundation turns it into what most people know as Linux "forming both the full operating system and the core of most Linux distributions". ***Distributions*** are the versions of Linux, GNU programs, and other tools that are offered by different companies, organizations, or individuals. Popular distributions include **Red Hat, Debian, SuSE, Caldera**, and others. Each distribution might be based on a different version of the Linux kernel, but all migrate forward over time, picking up core changes that are made to the kernel and keeping everything in somewhat loose synchronization.

Eric S. Raymond's famous essay, "The Cathedral and the Bazaar," argues that most commercial software is built like cathedrals by small groups of artisans working in isolation. Open-source software, like Linux, is developed collectively over the Internet, which serves as an electronic bazaar for innovative ideas. The first of the two programming styles is closed source - the traditional factory-production model of proprietary software, in which customers get a sealed block of computer binary that they cannot examine, modify, or evolve. The other style is open-source, the Internet engineering tradition in which software source code is generally available for inspection, independent peer review, and rapid evolution. Linux operating environment is the standard-bearer of the open source approach.

With Open Source products like Linux, new changes come through an open development model, meaning that all new versions are available to the public, regardless of their quality. "Linux' s versioning scheme is designed to let users understand whether they're using a stable version or a development version," says Jim Enright, director of Oracle's Linux program office. "Even decimal-numbered releases [such as 2.0, 2.2, and 2.4] are considered stable versions, while odd-numbered releases [such as 2.3 and 2.5] are beta-quality releases intended for developers only."

For much of the 1990s, Linux was primarily an experiment: something that developers fiddled with and used on local servers to see how well it worked and how secure it was. Then, with the internet boom of the late 1990s, many companies started using Linux for their Web servers, fueling the first wave of corporate Linux adoption leading to over 30 percent penetration of web server market by 2002.

Open source movement today is no more about just Linux, there are hundreds of thousands of software products being worked on in the Free/Open Source Software (FOSS) mode—*Apache, MySQL, Postgres, Firefox ,Jboss* etc are some of the other popular members of this growing family who have proved themselves in the real world. Also, its influence is no more confined to CSE/IT areas alone; Open Source solutions are today available for many of the simulation, computing, design and visualization needs of the entire Scientific and Engineering community.

It is for reasons such as the above that most of the major global players in the computing arena such as IBM, Intel, Oracle, HP etc., all have started their own in-house FOSS initiatives.

What Makes Linux So Popular?

Here, we find a few of the important reasons.

It's Free

Linux is free. Really and truly free. One can browse to any of the distributors of Linux , find the "download" link and download a complete copy of the entire operating system plus extra software without paying any thing. One can also buy a boxed version of course. For a nominal price, the CDs are available, manuals get door-delivered, plus there is telephone or online support. By comparison, "home" versions of popular commercial OS would cost thousands of rupees.

With Linux you also don't have to worry about paying again every time you upgrade the operating system - the upgrades are obviously free too. With commercial OS, upgrades also have to be paid for every time one is announced.

It's Open Source

This means two things: First, that the CDs (or the download site) contain an entire copy of the source code for Linux. Secondly, the user can legally make modifications to improve it.

While this might not mean much to non- programmers, there are thousands of people with programming capability who could improve the code or fix problems quickly. When a problem is found, it is sent off to the coordinating team in charge of the module in question, who will update the software and issue a patch. What all this boils down to is that bugs in Linux get fixed much faster than any other operating system.

It's Modular

Commercial Operating Systems normally get installed as a complete unit. One cannot, for example, install them without their Graphical User Interface, or without its printing support — install everything or nothing.

Linux, on the other hand, is a very modular operating system. One could install or run exactly the bits and pieces of Linux that are needed. In most cases, the choice is on one of the predefined setups from the installation menu, but is not compulsory. In some cases this makes a lot of sense. For example, while setting up a server, one might want to disable the graphical user interface once it is set up correctly, thus freeing up memory and the processor for the more important task at hand.

It also allows the users to upgrade parts of the operating system without affecting the rest. For example, one could get the latest version of Gnome or KDE without changing the kernel.

It's got More Choices

Also due to its modularity, there is more choice of components to use. One example is the user interface. Many users choose KDE, which is very easy to learn for users with Windows experience. Others choose Gnome, which is more powerful but less similar to Windows. There are also several simple alternatives for less-powerful computers, which make less demands on the hardware available.

It's Portable

Linux runs on practically every piece of equipment which qualifies as a computer. It can be run on huge multiprocessor servers or a PDA. Apart from Pentiums of various flavors, there are versions of Linux (called "ports") on Atari, Amiga, Macintosh, PowerMac, PowerPC, NeXT, Alpha, Motorola, MIPS, HP, PowerPC, Sun Sparc, Silicon Graphics, VAX/MicroVax, VME, Psion 5, Sun UltraSparc, etc.

It's got lots of Extras

Along with the Linux CD, normally quite a lot of software gets thrown in, which is not usually included with operating systems. Using only the applications that come with Linux, one could set up a full web, ftp, database and email server for example. There is a firewall built into the kernel of the operating system, one or more office suites, graphics programs, music players, and lots more. Different distributions of Linux offer different "extra programs". Slackware, for example, is quite simple (though it still provides all the commonly needed programs), while SuSE Linux comes with seven CDs and a DVD-ROM!

It is Stable

All applications can crash, but in many systems, the only recourse is to switch off and reboot (and with some new "soft-switch" PCs, even that doesn't work - you have to pull out the power cable).

In comparison, Linux is rock-solid. Every application runs independently of all others - if one crashes, it crashes alone. Most Linux servers run for months on end, never shutting down or rebooting. Even the GUI is independent of the kernel of the operating system.

It's got Networking

The networking facilities offered by Linux are positively awe-inspiring. One can use terminal sessions, secure shells, share drives from across the world, run a wide variety of servers and much more. The user can, for example, connect XWindows to another Linux PC across a network. If there is more than one computer, one does not have to physically use the screen, keyboard and mouse connected to each computer - from any computer connect to any other computer, running applications etc. as if they were on the local system.

Multiple OS' s on a PC

Dual options like having Windows as well as Linux are possible and one could select which one of them to load every time you switch on. Linux can read Windows' files - it supports the FAT and FAT32 file systems, and sometimes NTFS, so it's quite easy to transfer files from one operating system to the other. The opposite, however, is not possible.

Generally, Windows applications cannot run under Linux, though there is a module called WINE which runs various small Windows programs in Linux. However, Open Office – an open source product loaded on to the Linux system— can read and write MS-Office files. There are also other office suite options like Star Office, KOffice, GnomeOffice, WordPerfect Office, etc.

Can Windows and Linux machines interact via network? Definitely. You can use SAMBA to share files or connect to shared directories or printers. With SAMBA, the Linux computer could be set up function as a full NT server - complete with authentication, file/printer sharing and so on. Apart from that, Linux comes with excellent FTP, Web and similar services which are accessible to all computers.

Linux Configuration Tool

LinuxConf is a popular utility which allows the configuration of most parts of Linux and its applications from one place.

Salient Features of Linux

Here are some of the benefits and features that Linux provides over single-user operating systems and other versions of UNIX for the PC.

- **Full multitasking and 32-bit support** : Linux, like all other versions of UNIX, is a real multitasking system, allowing multiple users to run many programs on the same system at once. The performance of a 50 MHz 486 system running Linux is comparable to many low- to medium-end workstations, such as those from Sun Microsystems and DEC, running proprietary versions of UNIX. Linux is also a full 32-bit operating system, utilizing the special protected-mode features of the Intel 80386 and 80486 processors.
- **GNU software support** : Linux supports a wide range of free software written by the GNU Project, including utilities such as the GNU C and C++ compiler, gawk, groff, and so on. Many of the essential system utilities used by Linux are GNU software.
- **The X Window System** : The X Window System is the de facto industry standard graphics system for UNIX machines. A free version of The X Window System (known as "Xfree86") is available for Linux. The X Window System is a very powerful graphics interface, supporting many applications. For example, one can have multiple login sessions in different windows on the screen at once. Other examples of X Windows applications are Seyon, a powerful telecommunications program; Ghostscript, a

PostScript language processor; and XTetris, an X Windows version of the popular game.

- **TCP/IP networking support** : TCP/IP ("Transmission Control Protocol/Internet Protocol") is the set of protocols which links millions of computers into a worldwide network known as the Internet. With an Ethernet connection, one can have access to the Internet or to a local area network from your Linux system. Or, using SLIP ("Serial Line Internet Protocol"), you can access the Internet over the phone lines with a modem.
- **Virtual memory and shared libraries**: Linux can use a portion of the hard drive as virtual memory, expanding the total amount of available RAM. Linux also implements shared libraries, allowing programs which use standard subroutines to find the code for these subroutines in the libraries at runtime. This saves a large amount of space, as each application doesn't store its own copy of these common routines.

Linux Distributions

Here are some of the more popular distributions of Linux.

- Mandrake
- Red Hat
- SuSE
- Caldera
- Corel
- Debian
- Slackware
- TurboLinux

1.2 INTRODUCTION TO UNIX/LINUX FILE SYSTEM

Files are stored on devices such as hard and floppy disks. O/S defines a *file system* on the devices. Many O/S use a *hierarchical* file system (See Figure 1.5).

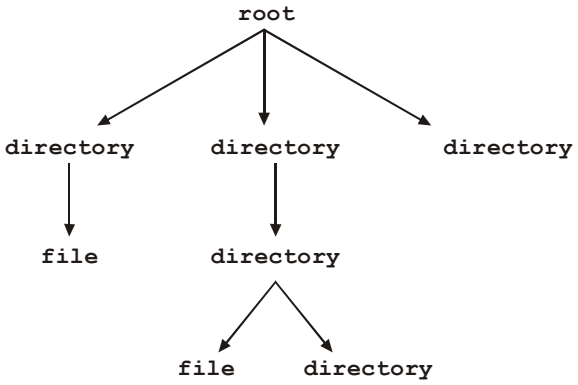

Figure 1.5 Hierarchical File System.

A directory is a file that keeps a list of other files. This list is the set of children of that directory node in the file system. A directory cannot hold any other kind of data.

On MSDOS a file system resides on each floppy or partition of the hard disk. The device name forms part of the file name.

On UNIX there is a single file system. Devices are *mounted* into this file system. (Use the command mount for this.)

File and Directory Naming

An individual node of the file system has its own name. Naming conventions differ between O/S's. In MSDOS, a name is constructed of up to 8+3 characters. Windows95 uses some tricks on top of the MSDOS file system to give "long file names" of up to 255 characters. In "standard UNIX" (POSIX) a name may consist of up to 256 characters.

The full file names are constructed by concatenating the directory names from the root down to the file, with some special separator between names. This is known as absolute path naming. In MSDOS, the full path name also includes the drive name.

Example : MSDOS
C:\expsys\lectures\search.txt

Example : UNIX
/usr/usrs/os
/usr/usrs/os/myfile

Relative naming means that files are named from some special directory:

. current directory (UNIX and MSDOS)

.. parent directory (UNIX and MSDOS)

~ home directory (some UNIX shells)

~user home directory of user (some shells)

Example : UNIX

~nbv/../rao/dir1/./../file1

If just the name itself is given without any special prefixes (such as /, ., .., ~) then it refers to the file in the current working directory.

Linux Directory Tree

Like all good operating systems, Linux allows the privilege of storing information indefinitely (or at least until the next disk crash) in abstract data containers called files. The organization, placement and usage of these files come under the general umbrella of the file hierarchy. As a system administrator, we need to be very familiar with the file hierarchy. To maintain the system, install software and manage user accounts we have to have better idea of it.

A typical Linux host's root directory contain:

bin	etc	lost+found	root	usr
boot	home	mnt	sbin	var
dev	lib	proc	tmp	

Why was it done like this?

Historically, the location of certain files and utilities has not always been standard (or fixed). This has lead to problems with development and upgrading between different "distributions" of Linux. The Linux directory structure (or file hierarchy) was based on

existing flavors of UNIX, but as it evolved, certain inconsistencies developed. These were often small things like the location (or placement) of certain configuration files, but it resulted in difficulties porting software from host to host.

To combat this, a file standard was developed. This is an evolving process, to date resulting in a fairly static model for the Linux file hierarchy. Here, we will examine how the Linux file hierarchy is structured, how each component relates to the overall OS and why certain files are placed in certain locations.

The root

The top level of the Linux file hierarchy is referred to as the root (or /) [Kernigham]. The root directory typically contains several other directories including the following given in Table 1.1.

Table 1.1 Major UNIX Directories

Directory	Contains
bin/	Required Boot-time binaries
boot/	Boot configuration files for the OS loader and kernel image
Dev/	Device files
etc/	System configuration files and scripts
home/	User/Sub branch directories
lib/	Main OS shared libraries and kernel modules
lost+found/	Storage directory for "recovered" files
mnt/	Temporary point to connect devices to
proc/	Pseudo directory structure containing information about the kernel, currently running processes and resource allocation
root/	Linux (non-standard) home directory for the root user. Alternate location being the / directory itself
sbin/	System administration binaries and tools
tmp/	Location of temporary files
usr/	Difficult to define - it contains almost everything else including local binaries, libraries, applications and packages (including X Windows)
var/	Variable data, usually machine specific. Includes spool directories for mail and news

Generally, the root should not contain any additional files - it is considered bad form to create other directories off the root, nor should any other files be placed there.

Why root?

The name "root" is based on the analogous relationship between the UNIX files system structure and a tree! Quite simply, the file hierarchy is an inverted tree.

Every part of the file system eventually can be traced back to one central point, the root. The concept of a "root" structure has now been (partially) adopted by other operating systems such as Windows NT. However, unlike other operating systems, UNIX doesn't

have any concept of "drives". While this will be explained in detail in a later chapter, it is important to be aware of the following:

The file system may be spread over several physical devices; different parts of the file hierarchy may exist on totally separate partitions, hard disks, CD-ROMs, network file system shares, floppy disks and other devices.

This separation is transparent to the file system hierarchy, user and applications. Different "parts" of the file system will be "connected" (or mounted) at startup; other parts will be dynamically attached as required.

In the following pages, we examine some of the most important directory structures in the Linux file hierarchy.

/home: Home for all users

The /home directory structure contains the home directories for most login-enabled users (some notable exceptions being the root user and (on some systems) the www/web user). While most small systems will contain user directories directly off the /home directory (for example, /home/jamiesob), on larger systems is common to subdivide the home structure based on classes (or groups) of users, for example:

```
/home/admin          # Administrators
/home/finance        # Finance users
/home/humanres       # Human Resource users
/home/mgr            # Managers
/home/staff          # Other people
```

One must be extremely careful when allowing a user to have a home directory in a location other than the /home branch. The problem occurs when the system administrator has to back-up the system - it is easy to miss a home directory if it isn't grouped with others in a common branch such as /home.

/root is the home directory for the root user. That is, when he log's in with username as root he will be placed in his directory. If, for some strange reason, the /root directory doesn't exist, then the root user will be logged in the / directory - this is actually the traditional location for root users.

It is advisable that a system administrator should **never** use the root account for day to day user-type interaction; the root account should only be used for system administration purposes only.

/usr and /var

It is often slightly confusing to see that /usr and /var both contain similar directories:

```
/usr
        X11R6          games            libexec        src
        bin            i486-linux-libc5 local          tmp
        dict           include          man
        doc            info             sbin
        etc            lib              share

/var
        catman    local    log      preserve    spool
        lib       lock     nis      run         tmp
```

It becomes even more confusing when you start examining the maze of links which intermingle the two major branches.

To put it simply, /var is for **VAR**iable data/files. /usr is for **US**e**R** accessible data, programs and libraries. Unfortunately, history has confused things - files which should have been placed in the /usr branch have been located in the /var branch and vice versa. Thus to "correct" things, a series of links have been put in place.

The following are a few highlights of the /var and /usr directory branches:

/usr/local

All software that is installed on a system after the operating system package itself should be placed in the /usr/local directory. Binary files should be located in the /usr/local/bin (generally /usr/local/bin should be included in a user's PATH setting). By placing all installed software in this branch, it makes backups and upgrades of the system far easier - the system administrator can back-up and restore the entire /usr/local system with more ease than backing-up and restoring software packages from multiple branches (i.e.. /usr/src, /usr/bin etc.).

An example of a /usr/local directory is listed below:

bin	games	lib	rsynth	cern
man	sbin	volume-1.11	info	
mpeg	speak	www	etc	java
netscape	src			

As you can see, there are a few standard directories (bin, lib and src) as well as some that contain installed programs.

Linux is a very popular platform for C/C++, Java and Perl program development. As we will discuss in later chapters, Linux also allows the system administrator to actually modify and recompile the kernel. Because of this, compilers, libraries and source directories are treated as "core" elements of the file hierarchy structure.

The /usr structure plays host to three important directories:

/usr/include holds most of the standard C/C++ header files - this directory will be referred to as the primary include directory in most Makefiles.

/usr/lib holds most static libraries as well as hosting subdirectories containing libraries for other (non C/C++) languages including Perl and TCL. It also plays host to configuration information for ldconfig.

/usr/src holds the source files for most packages installed on the system. This is traditionally the location for the Linux source directory (/usr/src/linux), for example:

linux linux-2.6.31 redhat.

/var/spool

This directory has the potential for causing a system administrator a bit of trouble as it is used to store (possibly) large volumes of temporary files associated with printing, mail and news. /var/spool may contain something like:

at	lp	lpd	mqueue	samba	uucppublic
cron	mail	rwho	uucp		

In this case, there is a printer spool directory called lp (used for storing print request for the printer lp) and a /var/spool/mail directory that contains files for each user's incoming mail.

Keep an eye on the space consumed by the files and directories found in /var/spool. If a device (like the printer) isn't working or a large volume of e-mail has been sent to the system, then much of the hard drive space can be quickly consumed by files stored in this location.

/var/log

Linux maintains a particular area in which to place logs (or files which contain records of events). This directory is /var/log.

This directory usually contains:

cron	lastlog	maillog.2	samba-log.	secure.2	uucp
cron.1	log.nmb	messages	samba.1	sendmail.st	wtmp
cron.2	log.smb	messages.1	samba.2	spooler	xferlog
dmesg	maillog	messages.2	secure	spooler.1	xferlog.1
httpd	maillog.1	samba	secure.1	spooler.2	xferlog.2

/usr/X11R6

X-Windows provides UNIX with a very flexible graphical user interface. Tracing the X Windows file hierarchy can be very tedious, especially when we are trying to locate a particular configuration file or trying to remove a stale lock file.

Most of X Windows is located in the /usr structure, with some references made to it in the /var structure.

Typically, most of the action is in the /usr/X11R6 directory (this is usually an alias or link to another directory depending on the release of X11 - the X Windows manager). This will contain:

bin doc include lib man

The main X Windows binaries are located in /usr/X11R6/bin. This may be accessed via an alias of /usr/bin/X11.

Configuration files for X Windows are located in /usr/X11R6/lib. To really confuse things, the X Windows configuration utility, xf86config, is located in /usr/X11R6/bin, while the configuration file it produces is located in /etc/X11 (XF86Config)!

Because of this, it is often very difficult to get an "overall picture" of how X Windows is working - my best advice is read up on it before you start modifying (or developing with) it.

bin's

A very common mistake amongst first time UNIX users is to incorrectly assume that all "bin" directories contain temporary files or files marked for deletion. This misunderstanding comes about because:

- People associate the word "bin" with rubbish
- Some unfortunate GUI based operating systems use little icons of "trash cans" for the purposes of storing deleted/temporary files.

However, bin is short for binary - binary or executable files. There are four major bin directories (none of which should be used for storing junk files :)

- /bin
- /sbin

- /usr/bin
- /usr/local/bin

All of the bin directories serve similar but distinct purposes; the division of binary files serves several purposes including ease of backups, administration and logical separation. Note that while most binaries on Linux systems are found in one of these four directories, not all are.

/bin

This directory must be present for the OS to boot. It contains utilities used during the startup; a typical listing would look something like:

mail	df	gzip	mount	stty
arch	dialog	head	mt	su
ash	dircolors	hostname	mt-GNU	sync
bash	dmesg	ipmask	mv	tar
cat	dnsdomainname	kill	netstat	tcsh
chgrp	domainname	killall	ping	telnet
chmod	domainname-yp	ln	ps	touch
chown	du	login	pwd	true
compress	echo	ls	red	ttysnoops
cp	ed	mail	rm	umount
cpio	false	mailx	rmdir	umssync
csh	free	mkdir	setserial	uname
cut	ftp	mkfifo	setterm	zcat
date	getoptprog	mknod	sh	zsh
dd	gunzip	more	sln	

Note that this directory contains the shells and some basic file and text utilities (ls, pwd, cut, head, tail, ed etc). Ideally, the /bin directory will contain as few files as possible as this makes it easier to take a direct copy for recovery boot/root disks.

/sbin

/sbin Literally "System Binaries". This directory contains files that should generally only be used by the root user, though the Linux file standard dictates that no access restrictions should be placed on normal users to these files. It should be noted that the PATH setting for the root user includes /sbin, while it is (by default) not included in the PATH of normal users.

The /sbin directory should contain essential system administration scripts and programs, including those concerned with user management, disk administration, system event control (restart and shutdown programs) and certain networking programs.

As a general rule, if users need to run a program, then it should not be located in /sbin. A typical directory listing of /sbin looks like :

adduser	ifconfig	mkfs.minix	rmmod
agetty	init	mklost+found	rmt
arp	insmod	mkswap	rootflags
badblocks	installpkg	mkxfs	route
bdflush	kbdrate	modprobe	runlevel

chattr	killall5	mount	setup
clock	ksyms	netconfig	setup.tty
debugfs	ldconfig	netconfig.color	shutdown
depmod	lilo	netconfig.tty	swapdev
dosfsck	liloconfig	pidof	swapoff
dumpe2fs	liloconfig-color	pkgtool	swapon
e2fsck	lsattr	pkgtool.tty	telinit
explodepkg	lsmod	plipconfig	tune2fs
fdisk	makebootdisk	ramsize	umount
fsck	makepkg	rarp	update
fsck.minix	mkdosfs	rdev	vidmode
genksyms	mke2fs	reboot	xfsck
halt	mkfs	remov	pkg

We should note that :

/usr/sbin - used for non-essential admin tools.

/usr/local/sbin - locally installed admin tools.

/usr/bin

This directory contains most of the user binaries - in other words, programs that users will run. It includes standard user applications including editors and email clients as well as compilers, games and various network applications.

/usr/local/bin

To this point, we have examined directories that contain programs that are (in general) part of the actual operating system package. Programs that are installed by the system administrator after that point should be placed in /usr/local/bin. The main reason for doing this is to make it easier to back up installed programs during a system upgrade, or in the worst case, to restore a system after a crash.

/etc is one place where the root user will spend a lot of time. It is not only the home to the all important passwd file, but contains just about every configuration file for a system (including those for networking, X Windows and the file system).

The /etc branch also contains the skel, X11 and rc.d directories.

/etc/skel contains the skeleton user files that are placed in a user's directory when their account is created.

/etc/X11 contains configuration files for X Windows.

/etc/rc.d is contains rc directories - each directory is given by the name rcn.d (n is the run level) - each directory may contain multiple files that will be executed at the particular run level. A sample listing of a /etc/rc.d directory looks something like:

init.d	rc.local	rc0.d	rc2.d	rc4.d	rc6.d
rc	rc.sysinit	rc1.d	rc3.d	rc5.d	

/proc

The /proc directory hierarchy contains files associated with the *executing* kernel. The files contained in this structure contain information about the state of the system's resource

usage (how much memory, swap space and CPU is being used), information about each process and various other useful pieces of information.

/dev

We will be discussing /dev in detail in the next chapter, however, for the time being, you should be aware that this directory is the primary location for special files called **device files**.

1.3 MAIN PAGES

All the UNIX command information is organized in a special fashion like the following.

- The user-level commands are all in Section One.
- Section Two is the UNIX Application Programmer's Interface, API (i.e. C functions directly supported by UNIX).
- Section Three is library extensions to these.
- Section Four defines devices known to UNIX.
- Section Five defines common file formats.
- Sections Local and New are for stuff we have added to our local system.

If we run **man** command with a name first it will check for commands with that name.

Example:

man sleep

man 2 sleep

This displays details of sleep library function if available

man 3 sleep

This displays details of sleep system call if exists any.

The **apropos** command can be used to displays names of all the commands whose manual page contains a search pattern.

Example :

apropos TERM

This displays names of the UNIX commands, system calls or library functions whose manual page contains the search pattern TERM.

We do have another command in recent Linux distribution known as whatis. It requires a command name as argument and displays short description of the command. An example usage is:

whatis ls

(For example, we have used ls. One can use any other thing).

However, it displays meaningful information if already system contains whatis database. Otherwise, system administrator has to create the same using /usr/sbin/makewhatis command.

Of course, -f option with man command works like whatis command. Check the output of the following commands:

whatis ls

man –f ls

Also, -k option with man command works like apropos command. Verify the output of the following commands:

apropos ls

man –k ls

The info command

All Linux systems and other UNIX derivatives which use GNU utilities will be having another means of having online help to the users by executing info command. Here, the manual files are stored in files known as INFO FILES and are organized in a specific manner using texinfo or makeinfo commands. Conceptually, this info service is different from manual pages through man command in the following manner.

1. The info command supports emacs style editor interface with which we can browse the online manuals in a better manner compared to man command.
2. The manual pages will be having links to other manual pages like our web pages.
3. This is little complex organization than man pages organization.

We can start info by just simply typing info at the command prompt like:

info

We will get a screen as shown in Figure 1.6 . We can browse through the info screens and know details of the commands of our interest.

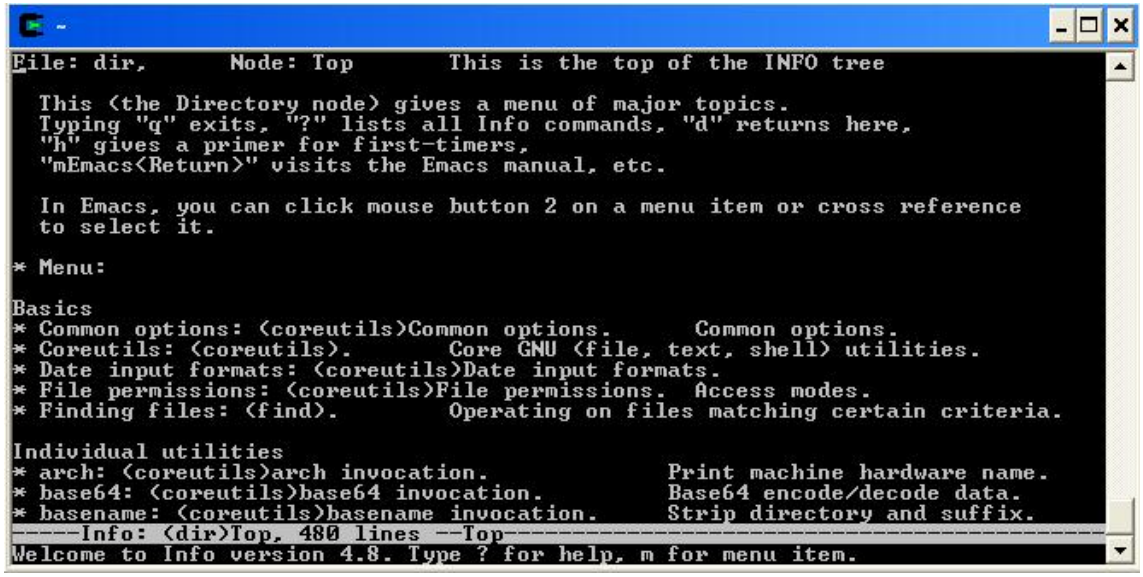

Figure 1.6 Sample Info screen

We can also see the online manual pages of a specific command by executing info command with the command name as argument like:

info ls

We will get a info screen as shown in Figure 1.7.

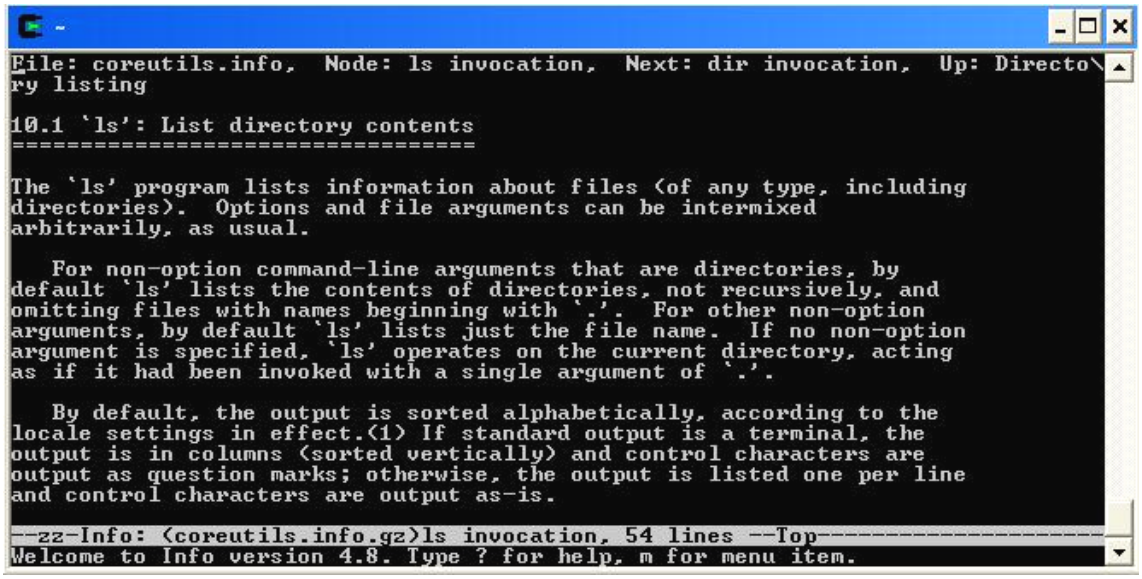

Figure 1.7 Another Info screen

1. 4 THE FIRST COMMAND 'CAT '

In order to login from UNIX/Linux machines, you have to first get username and password. Approach your system administrator. Once you get them, power on the machine and you may find boot loader options from which you can select Linux or UNIX. Wait for a while and if possible go through the system messages appearing on the screen. After loading all the necessary drivers you may find 'login' prompt either in character mode or in graphical mode. Now enter your username and password.

First, one may be interested to create a file or view a file. The simplest command available in UNIX system to create and view files is 'cat'. Follow the following exercise to create and see the files.

To create files, one can follow the following exercise at the command or dollar prompt.

Example:

```
cat > ABC
This is a test file.
I wish you find happy to create first file.
^d
```

This is also used to see the file(s) content. If the file contains more lines then it simply scrolls the matter of that file.

Example:

cat ABC

This command is also used to create duplicate of a file.

Example:

cat ABC >XYZ

or

cat <ABC >XYZ

XYZ becomes duplicate copy of file the file ABC.

This cat command can be used to see the content of more than one file. The following example displays the content of the files ABC and XYZ.

Example:

cat ABC XYZ

This cat command can be used to join the content of two or more files and create another file.

Example:

cat ABC XYZ > MNO

Now MNO file contains the content of both the files XYZ and ABC.

While joining two or more files to create a combined file we can add interactive input also. For example, assuming that the files ABC, XYZ and PQR contents as given below; we can create combined file along some extra interactive input between file XYZ and PQR by executing the following command.

Example:

cat ABC XYZ - PQR > PPP

Now file PPP contains content of ABC, XYZ, the interactive input and the content of file PQR in the same order. By changing the location of -, we can add interactive input between any two files.

For example the following command creates a file PPP1 such that it contains content of the file ABC, interactive input, content of the files PQR and XYZ.

cat ABC - PQR XYZ > PPP1

We can see the output by executing the command **cat PPP1**.

We can also add interactive input between any two files while joining the files. For example, in the following command we are adding interactive input between the files A & B, B & C, and C & D. Each time, we go on type some thing and followed by CTRL D will be used.

cat A – B – C – D >PPP2

See the output of an interactive session on the terminal.

$cat ABC
To,
Mr. NTR
Heaven
$cat XYZ
From,
Dr.NB Venkateswarlu
47-15-66/44
Dwarakanagar
Visakhapatnam – 530016
$cat PQR
Yours Sincerely
Dr.N.B. Venkateswarlu
$cat ABC XYZ – PQR >PPP
Dear Sir,
There is a great vacuum in Indian politics. We strongly believe that you are the best possible candidate for the position of PM. We request you to have your re-incarnation on the Indian soil and take over the country. Looking forward for your arrival.

Thanking You Sir.
^d
$cat PPP
To,
Mr. NTR
Heaven
From,
Dr.NB Venkateswarlu
47-15-66/44
Dwarakanagar
Visakhapatnam – 530016
Dear Sir,
There is a great vacuum in Indian politics. We strongly believe that you are the best possible candidate for the position of PM. We request you to have your re-incarnation on the Indian soil and take over the country. Looking forward for your arrival.

Thanking You Sir.
Yours Sincerely
Dr.N.B. Venkateswarlu

Command editing in the tcsh

While working in the shell, we can use the following handy keyboard shortcuts. In fact, there exist many such shortcuts; however because of space constraints we are listing most useful ones.

Backspace	delete previous character
CTRL-d	delete next character
CTRL-k	delete rest of line
CTRL-a	go to start of line
CTRL-e	go to end of line
CTRL-b	go backwards without deleting
CTRL-f	go forward without deleting
TAB	complete filename or command up to the point of uniqueness
CTRL-u	cancel whole line
CTRL-p	show the last command typed, then the one before that, etc. (you can also use the cursor up key for this)
CTRL-n	go forwards in the history of commands (you can also use the cursor down key for this)
CTRL-c	cancel the processes after it has started
CTRL-z	suspend a running process (e.g. in order to do something else in between) you can then put the process in the background with **bg**
CTRL-l	redraws the screen
CTRL-x	Undo the effect of last editing command on the current line
CTRL-k	Kill the text from the current cursor position to the end of the line.
ALT-d	Kill from the cursor to the end of the current word, or, if between words, to the end of the next word. Word boundaries are the same as those used by M-f.
ALT-<DEL>	Kill from the cursor the start of the current word, or, if between words, to the start of the previous word. Word boundaries are the same as those used by M-b.
CTRL-w	Kill from the cursor to the previous whitespace. This is different than M-<DEL> because the word boundaries differ. Here is how to *yank* the text back into the line. Yanking means to copy the most-recently-killed text from the kill buffer.
CTRL-y	Yank the most recently killed text back into the buffer at the cursor.
ALT-y	Rotate the kill-ring, and yank the new top. You can only do this if the prior command is C-y or M-y.

1.5 CONCLUSIONS

We have discussed about basics of operating systems with specific emphasis to UNIX/ Linux. The popular acceptance of Linux is explained along with its relative benefits. UNIX directory structure is explained in detail. Basic commands such as man and cat are explained with live examples.