MCA Part-II

Paper-XIII: Operating System

Topic: UNIX

PREPARED BY: DR. KIRAN PANDEY

(School of Computer Science)

Email-id: kiranpandey.nou@gmail.com

INTRODUCTION

An Operating System (OS) is an interface between a computer user and computer hardware. It is a system software. It performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

Some popular Operating Systems include UNIX, Linux, Windows, OS X, VMS, OS/400, AIX, z/OS, etc.

UNIX is an operating system which was first developed in the 1960s, and has been under constant development ever since. By operating system, we mean the suite of programs which make the computer work. It is a stable, multi-user, multi-tasking system for servers, desktops and laptops. The following are the uses of the UNIX operating system:

- Universally used for high-end number crunching applications.
- Wide use in CAD/CAM arena.
- Ideally suited for scientific visualization.
- Preferred OS platform for running internet services such as WWW, DNS, DHCP, NetNews, Mail, etc., due to networking being in the kernel for a long time now.

- Easy access to the core of the OS (the kernel) via C, C++, Perl, etc.
- Stability of the operating environment.
- The availability of a network extensible window system.
- The open systems philosophy.

BASIC FEATURES OF UNIX OS

The fundamental features which made UNIX such a phenomenally successful operating systems are:

- Portable: It is written in high-level language, 'C' making it easy to port to different configurations.
- Rich and productive programming environment: It is a good operating system, especially, for programs. It provides features that allow complex programs to be built from simpler programs.
- Consistent format for files: the byte stream, makes application programs easier to write.
- Multiuser: It is a multi-user, multi-process system. Each user can execute several processes simultaneously.
- It hides the machine architecture from the user, making it easier to write programs that run on different hardware implementation.
- Multi-tasking: More than one program can be run at a time.
- Tools for program development: Supports a wide range of support tools (debuggers, compilers).

STRUCTURE OF UNIX OS

UNIX is a layered operating system. The innermost layer is the hardware that provides the services for the OS. The following are the components of the UNIX OS.

The Kernel

The kernel is the heart of the operating system. It interacts with the hardware and does most of the tasks like memory management, task scheduling and file management. It also provides the services to the user programs. These user programs don't need to know anything about the hardware. They just need to know how to interact with the kernel and it's up to the kernel to provide the desired service. Since UNIX has been written in high level language C, user programs are independent of the underlying hardware, making them readily portable to new systems.

User programs interact with the kernel through a set of standard system calls. These system calls request services to be provided by the kernel. Such services would include accessing a file: open close, read, write, link, or execute a file; starting or updating accounting records; changing ownership of a file or directory; changing to a new directory; creating, suspending, or killing a process; enabling access to hardware devices; and setting limits on system resources.

The kernel’s job to keep each process and user separate and to regulate access to system hardware, including CPU, memory, disk and other I/O devices.

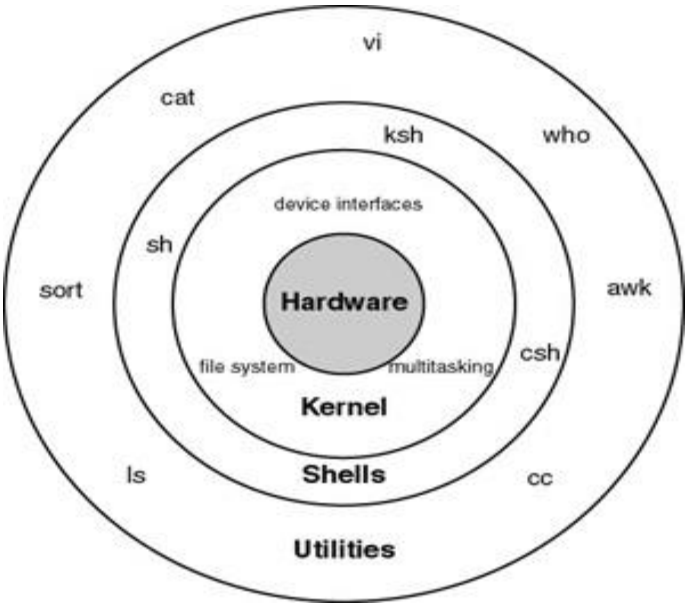

Figure 1: UNIX structure

The Shell

The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want. The shell is often called a command line shell, since it presents a single prompt for the user. The user types a command; the shell invokes that command, and then presents the prompt again when the command has finished. This is done on a line-by-line basis, hence the term "command line".

The shell program provides a method for adapting each user's setup requirements and storing this information for re-use. The user interacts with /bin/sh, which interprets each command typed. Internal commands are handled within the shell (set, unset), external commands are cited as programs (ls, grep, sort, ps).

There are a number of different command line shells (user interfaces).

- Bourne (sh)
- Korn (krn)

- C shell (csh)

In Windows, the command interpreter is based on a graphical user interface.

In UNIX, there is a line-orientated command interpreter.

System Utilities

There are various commands and utilities which you can make use of in your day to day activities. **cp**, **mv**, **cat** and **grep**, etc. are few examples of commands and utilities. There are over 250 standard commands plus numerous others provided through 3rd party software. All the commands come along with various options. The system utilities are intended to be controlling tools that do a single task exceptionally well. Users can solve problems by integrating these tools instead of writing a large monolithic application program.

Like other UNIX flavours, Linux's system utilities also embrace server programs called daemons that offer remote network and administration services (e.g. telnetd provides remote login facilities, httpd serves web pages).

Files and Directories

All the data of UNIX is organized into files. All files are then organized into directories. These directories are further organized into a tree-like structure called the **filesystem**.

Application Programs

Some application programs include the emacs editor, gcc (a C compiler), g++ (a C++ compiler), xfig (a drawing package), latex (a powerful typesetting language).

UNIX works very differently. Rather than having kernel tasks examine the requests of a process, the process itself enters kernel space. This means that rather than the process waiting "outside" the kernel, it enters the kernel itself (i.e. the process will start executing kernel code for itself).

This may sound like a formula for failure, but the ability of a process to enter kernel space is strictly prohibited (requiring hardware support). For example, on x86, a

process enters kernel space by means of system calls - well known points that a process must invoke in order to enter the kernel.

When a process invokes a system call, the hardware is switched to the kernel settings. At this point, the process will be executing code from the kernel image. It has full powers to wreak disaster at this point, unlike when it was in user space. Furthermore, the process is no longer pre-emptible.

FILE SYSTEM IN UNIX

A file system is a logical collection of files on a partition or disk. A partition is a container for information and can span an entire hard drive if desired.

Your hard drive can have various partitions which usually contain only one file system, such as one file system housing the **/file system** or another containing the **/home file system**.

Everything in Unix is considered to be a file, including physical devices such as DVD-ROMs, USB devices, and floppy drives.

Unix uses a hierarchical file system structure, much like an upside-down tree, with root (/) at the base of the file system and all other directories spreading from there.

A Unix filesystem is a collection of files and directories that has the following properties –

- It has a root directory (**/**) that contains other files and directories.
- Each file or directory is uniquely identified by its name, the directory in which it resides, and a unique identifier, typically called an **inode**.
- By convention, the root directory has an **inode** number of **2** and the **lost+found** directory has an **inode** number of **3**. Inode numbers **0**and **1** are not used. File inode numbers can be seen by specifying the **-i option** to **ls command**.

- It is self-contained. There are no dependencies between one filesystem and another.

The constituents of UNIX file system can be one of the following types:

Ordinary files

Ordinary files can contain text, data, or program information. Files cannot contain other files or directories. UNIX filenames are not broken into a name part and an extension part, instead they can contain any keyboard character except for '/' and be up to 256 characters long. However, characters such as *,?,# and & have special meaning in most shells and should not therefore be used in filenames. Putting spaces in filenames also makes them difficult to manipulate, so it is always preferred to use the underscore '_'.

Directories

Directories are folders that can contain other files and directories. A directory is a collection of files and/or other directories. Because a directory can contain other directories, we get a directory hierarchy. The "top level" of the hierarchy is the root directory.

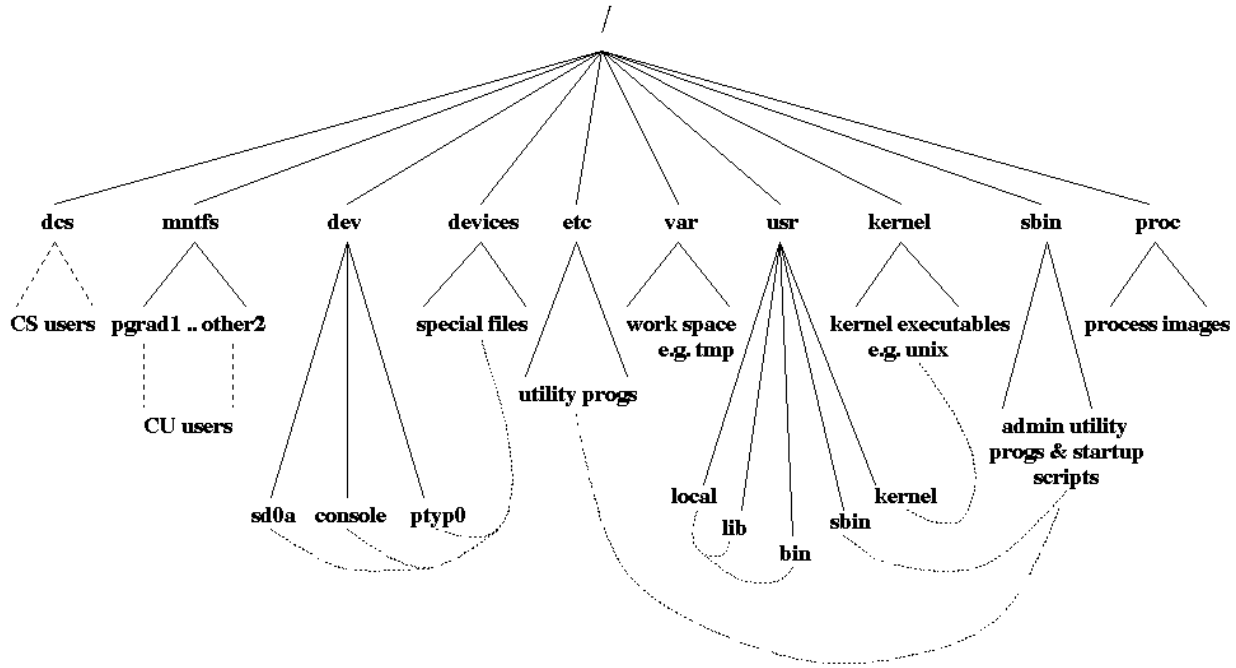

Figure 2: UNIX File System

The directories have specific purposes and generally hold the same types of information for easily locating files. Following are the directories that exist on the major versions of Unix –

S.No.	Directory & Description
1	**/** This is the root directory which should contain only the directories needed at the top level of the file structure
2	**/bin** This is where the executable files are located. These files are available to all users

3	**/dev** These are device drivers
4	**/etc** Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups, ethernet, hosts, where to send critical messages
5	**/lib** Contains shared library files and sometimes other kernel-related files
6	**/boot** Contains files for booting the system
7	**/home** Contains the home directory for users and other accounts
8	**/mnt** Used to mount other temporary file systems, such as **cdrom** and **floppy** for the **CD-ROM** drive and **floppy diskette drive**, respectively
9	**/proc** Contains all processes marked as a file by **process number** or other

	information that is dynamic to the system
10	**/tmp** Holds temporary files used between system boots
11	**/usr** Used for miscellaneous purposes, and can be used by many users. Includes administrative commands, shared files, library files, and others
12	**/var** Typically contains variable-length files such as log and print files and any other type of file that may contain a variable amount of data
13	**/sbin** Contains binary (executable) files, usually for system administration. For example, ***fdisk*** and ***ifconfig*** utlities
14	**/kernel** Contains kernel files

Links

A link is a pointer to another file. There are two types of links - a hard link to a file cannot be distinguished from the file itself. A soft link or also called the symbolic link provides an indirect pointer or shortcut to a file.

Each file is assigned an inode number by the kernel. Attributes in a file table in the kernel include its name, permissions, ownership, time of last modification, time of last access, and whether it is a file, directory or some other type of entity.

A -i flag to ls shows the inode number of each entry.

A -i flag to df shows the number of inodes instead of the amount of space.

The ls -l command lists the number of links to the inode in the second column. If you remove one of the names associated with a particular inode, the other names are preserved.

The file is not removed from the filesystem until the "link count" goes to zero. All permissions and ownerships of a link are identical to the file that you have linked to. You may not link to a directory.

You can form a symbolic link to a file by giving a -s flag to ln. The symbolic link has its own inode number. To remove a symbolic link to a directory, use rm, not rmdir.

File and Directory Permissions

Each file or a directory on a UNIX system has three types of permissions, describing what operations can be performed on it by various categories of users. These permissions are read (r), write (w) and execute (x) and the three categories of users are user or owner (u), group (g) and others (o). File ownership is an important component of Unix that provides a secure method for storing files. Every file in Unix has the following attributes –

- **Owner permissions** – The owner's permissions determine what actions the owner of the file can perform on the file.
- **Group permissions** – The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.

- **Other (world) permissions** – The permissions for others indicate what action all other users can perform on the file.

The file and directory permissions can only be modified by:

- their owners
- by the superuser (root)

by using the chmod(change [file or directory])mode system utility.

$ chmod options files

chmod accepts options in two forms. Firstly, permissions may be specified as a sequence of 3 octal digits. Each octal digit represents the access permissions for the user/owner, group and others respectively. The mapping of permissions onto their corresponding octal digits is as follows:

--- 0

--x 1

-w- 2

-wx 3

r-- 4

r-x 5

rw- 6

rwx 7

For example the command:

$ chmod 700 nou.txt

Sets the permissions on nou.txt to rwx------- (i.e., only the owner can read, write and execute the file). chmod also supports a -R option which can be used to recursively modify file permission by using chgrp (change group) .

$ chgrp group files

This command can be used to change the group that a file or directory belongs to.

File Compression and Backup

UNIX systems usually support a number of utilities for backing up and compressing files. The most useful are:

tar (tape archiver)

tar backs up entire directories and files onto a tape device or into a single disk file known as an archive.

To create a disk file tar archive, use

$ tar -cvf archivename filenames

where archivename will usually have .tar extension. Here the c option means create, v means verbose (output filenames as they are archived), and f means file.

- To list the contents of a tar archive, use:

  $ tar -tvf archivename

- To restore files from a tar archive, use:

  $ tar -xvf archivename

compress, gzip

compress and gzip are utilities for compressing and decompressing individual files (which may be or may not be archive files).

- To compress files, use:

  $ compress filename or $ gzip filename

- To reverse the compression process, use:

  $ compress -d filename or $ gzip -d filename

Handling Removable Media

Removable disks are prevalent in UNIX machines. CD-ROMs, DVD-ROMs, Zip disks, and floppies are all removable disks. When a UNIX system boots, normal filesystems are all mounted automatically. UNIX supports tools for accessing removable media such as CDROMs and floppy disks with the help of mount and umount commands.

The mount command serves to attach the file system found on some device to the file system tree. Conversely, the umount command will detach it again (it is very important to remember to do this when removing the floppy or CDROM). The file /etc/fstab contains a list of devices and the points at which they will be attached to the main filesystem:

$ cat /etc/fstab

/dev/fd0 /mnt/floppy auto rw,user,noauto 0 0

/dev/hdc /mnt/cdrom iso9660 ro,user,noauto 0 0

In this case, the mount point for the floppy drive is /mnt/floppy and the mount point for the CDROM is /mnt/cdrom. To access a floppy we can use:

$ mount /mnt/floppy

$ cd /mnt/floppy

$ ls

To force all changed data to be written back to the floppy and to detach the floppy disk from the file system, we use:

$ umount /mnt/floppy

Pipes and Filters

In UNIX systems you can connect two commands together so that the output from one program becomes the input of the next program. Two or more commands connected in this way form a pipe. Complex tasks can be performed by a series of commands one piping input into the next.

When a program takes its input from another program, it performs some operation on that input, and writes the result to the standard output. It is referred to as a ***filter***.

Sometimes, the use of intermediately files is undesirable. Consider an example where you are required to generate a report, but this involves a number of steps. First you have to concatenate all the log files. The next step after that is strip out comments, and then to sort the information. Once it is sorted, the results must be printed.

A typical approach is to do each operation as a separate step, writing the results to a file that then becomes an input to the next step.

Pipelining allows you to connect two programs together so that the output of one program becomes the input to the next program.

The symbol **|** (vertical bar) represents a pipe. Any number of commands can be connected in sequence, forming a pipeline .All programs in a pipeline execute at the same time.

The pipe (|) operator is used to create concurrently executing processes that pass data directly to one another. For example:

$ cat welcome.txt | sort | uniq

creates three processes (corresponding to cat, sort and uniq) that execute concurrently. As they execute, the output of the first process is passed on to the sort process which is in turn passed on to the uniq process. uniq displays its output on the screen (a sorted list of users with duplicate lines removed).

Redirecting Input and Output

Most UNIX system commands take input from your terminal and send the resulting output back to your terminal. A command normally reads its input from the standard input, which happens to be your terminal by default. Similarly, a command normally writes its output to standard output, which is again your terminal by default.

The output from programs is usually written to the screen, while their input usually comes from the keyboard (if no file arguments are given). In technical terms, we say that processes usually write to standard output (the screen) and take their input from standard input (the keyboard). There is in fact another output channel called standard error, where processes write their error messages; by default error messages are also sent to the screen.

To redirect standard output to a file instead of the screen, we use the > operator:

$ echo welcome

welcome

$ echo welcome > output

$ cat output

welcome

In this case, the contents of the file output will be destroyed if the file already exists. If instead we want to append the output of the echo command to the file, we can use the >> operator:

$ echo bye >> output

$ cat output

welcome bye

To capture standard error, prefix the > operator with a 2 (in UNIX the file numbers 0, 1 and 2 are assigned to standard input, standard output and standard error respectively), e.g.:

$ cat nonexistent 2>errors

$ cat errors

cat: nonexistent: No such file or directory

$

Standard input can also be redirected using the < operator, so that input is read from a file instead of the keyboard:

$ cat < output welcome

bye

You can combine input redirection with output redirection, but be careful not to use the same filename in both places. For example:

$ cat < output > output

will destroy the contents of the file output. This is because the first thing the shell does when it sees the > operator is to create an empty file ready for the output.

LIST OF IMPORTANT COMMAND IN UNIX

In the table below we summarize the more frequently used commands on a UNIX system. In this table, as in general, for most UNIX commands, **file**, could be an actual file name, or a list of file names, or input/output could be redirected to or from the command.

Table 1: UNIX COMMANDS

Command/Syntax	Description
awk/nawk [options] *file*	scan for patterns in a file and process the results
cat [options] *file*	concatenate (list) a file
cd [directory]	change directory
chgrp [options] *group file*	change the group of the file
chmod [options] *file*	change file or directory access permissions
chown [options] *owner file*	change the ownership of a file; can only be done by the superuser
chsh (**passwd -e/-s**) *username login_shell*	change the user's login shell (often only by the superuser)
cmp [options] *file1 file2*	compare two files and list where differences occur (text or binary files)
compress [options] *file*	compress file and save it as *file.Z*
cp [options] *file1 file2*	copy *file1* into *file2*; *file2* shouldn't already exist. This command creates or overwrites *file2*.
cut (options) [*file*(s)]	cut specified field(s)/character(s) from lines in file(s)
date [options]	report the current date and time
dd [if=infile] [of=outfile] [operand=value]	copy a file, converting between ASCII and EBCDIC or swapping byte order, as specified
diff [options] *file1 file2*	compare the two files and display the differences (text files only)
df [options] [resource]	report the summary of disk blocks and inodes free and in use
du [options] [*directory* or *file*]	report amount of disk space in use
echo [text string]	echo the text string to stdout
ed or **ex** [options] *file*	Unix line editors
emacs [options] *file*	full-screen editor
expr *arguments*	Evaluate the arguments. Used to do arithmetic, etc. in the

	shell.
file [options] *file*	classify the file type
find directory [options] [actions]	find files matching a type or pattern
finger [options] *user[@hostname]*	report information about users on local and remote machines
ftp [options] *host*	transfer file(s) using file transfer protocol
grep [options] 'search string' *argument* **egrep** [options] 'search string' *argument* **fgrep** [options] 'search string' *argument*	Search the argument (in this case probably a file) for all occurrences of the search string, and list them.
gzip [options] *file* **gunzip** [options] *file* **zcat** [options] *file*	Compress or uncompress a file. Compressed files are stored with a **.gz** ending
head [-number] *file*	display the first 10 (or number of) lines of a file
hostname	display or set (super-user only) the name of the current machine
kill [options] [-SIGNAL] [pid#] [%job]	Send a signal to the process with the process id number (pid#) or job control number (%n). The default signal is to kill the process.
ln [options] *source_file target*	link the *source_file* to the *target*
lpq [options] **lpstat** [options]	show the status of print jobs
lpr [options] *file* **lp** [options] *file*	print to defined printer
lprm [options] **cancel** [options]	remove a print job from the print queue
ls [options] [*directory* or *file*]	list *directory* contents or *file* permissions
mail [options] [user] **mailx** [options] [user] **Mail** [options] [user]	Simple email utility available on Unix systems. Type a period as the first character on a new line to send message out, question mark for help.

man [options] *command*	show the manual (**man**) page for a command
mkdir [options] *directory*	make a *directory*
more [options] *file* **less** [options] *file* **pg** [options] *file*	page through a text file
mv [options] *file1 file2*	move *file1* into *file2*
od [options] *file*	Octal dump a binary file, in octal, ASCII, hex, decimal, or character mode.
passwd [options]	set or change your password
paste [options] *file*	paste field(s) onto the lines in *file*
pr [options] *file*	filter the file and print it on the terminal
ps [options]	show status of active processes
pwd	print working (current) directory
rcp [options] *hostname*	remotely copy files from this machine to another machine
rlogin [options] *hostname*	login remotely to another machine
rm [options] *file*	remove (delete) a file or directory (**-r** recursively deletes the directory and its contents) (**-i** prompts before removing files)
rmdir [options] *directory*	remove a *directory*
rsh [options] *hostname*	remote shell to run on another machine
script *file*	saves everything that appears on the screen to file until **exit** is executed
sed [options] *file*	stream editor for editing files from a script or from the command line
sort [options] *file*	sort the lines of the *file* according to the options chosen
source *file* **.** *file*	Read commands from the *file* and execute them in the current shell. **source**: C shell, **.**: Bourne shell.
strings [options] *file*	Report any sequence of 4 or more printable characters ending in <NL> or <NULL>. Usually used to search binary files for ASCII strings.
stty [options]	set or display terminal control options
tail [options] *file*	display the last few lines (or parts) of a file
tar key[options] [*file*(s)]	Tape archiver--refer to man pages for details on creating, listing, and retrieving from archive files. Tar files can be stored on tape or disk.

tee [options] *file*	copy stdout to one or more files
telnet [host [port]]	communicate with another host using telnet protocol
touch [options] [date] *file*	create an empty file, or update the access time of an existing file
tr [options] *string1 string2*	translate the characters in string1 from stdin into those in string2 in stdout
uncompress *file.Z*	uncompress *file.Z* and save it as a file
uniq [options] *file*	remove repeated lines in a file
uudecode [*file*]	decode a uuencoded file, recreating the original file
uuencode [*file*] *new_name*	encode binary file to 7-bit ASCII, useful when sending via email, to be decoded as new_name at destination
vi [options] *file*	visual, full-screen editor
wc [options] [*file*(s)]	display word (or character or line) count for *file*(s)
whereis [options] *command*	report the binary, source, and man page locations for the command named
which *command*	reports the path to the command or the shell alias in use
who or **w**	report who is logged in and what processes are running
zcat *file.Z*	concatenate (list) uncompressed file to screen, leaving file compressed on disk

Questions

1. What are the important features of UNIX operating system?

2. Discuss brief history of UNIX.

3. Describe the structure of UNIX operating system.

4. How process management done in UNIX? Explain.

5. Discuss memory management in UNIX.

6. What is a file system in UNIX? What is its importance in UNIX?

7. What is a link in UNIX?

8. Discuss file and directory permissions in UNIX.

9. What are pipes and filters? Explain with the help of an example.

10. Discuss CPU scheduling in UNIX.