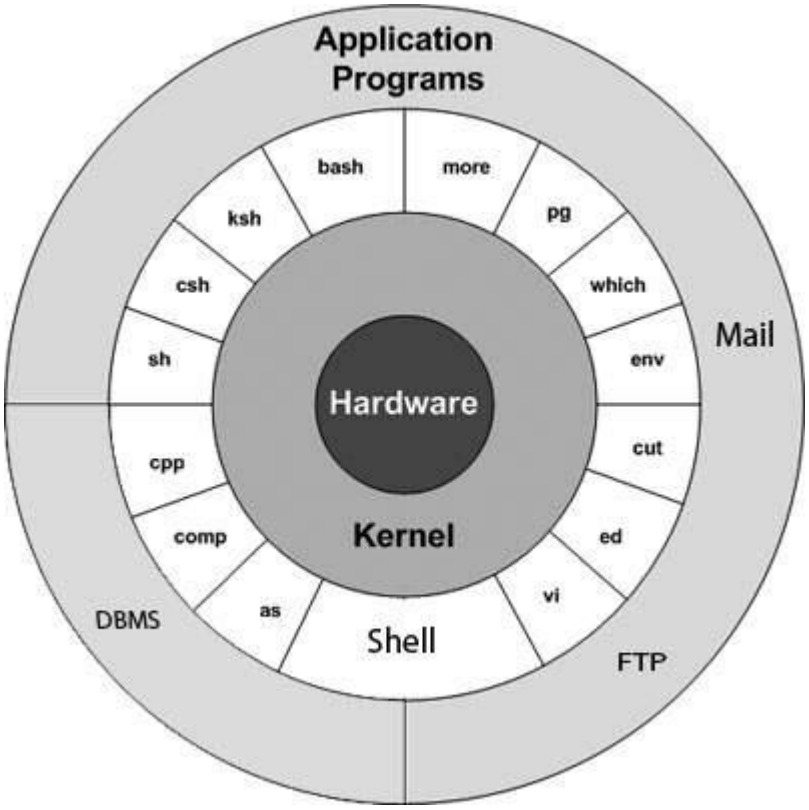
1 (a) With a neat diagram, explain UNIX architecture and its salient features. [07]

**Unix Architecture**

Here is a basic block diagram of a Unix system −




The main concept that unites all the versions of Unix is the following four basics −

- **Kernel** − The kernel is the heart of the operating system. It interacts with the hardware and most of the tasks like memory management, task scheduling and file management.
- **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 uses standard syntax for all commands. C Shell, Bourne Shell and Korn Shell are the most famous shells which are available with most of the Unix variants.
- **Commands and 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.

- **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**.

**Salient features of UNIX OS:**

Some key features of the Unix architecture concept are:

- Unix systems use a centralized operating system kernel which manages system and process activities.
- All non-kernel software is organized into separate, kernel-managed processes.
- Unix systems are preemptively multitasking: multiple processes can run at the same time, or within small time slices and nearly at the same time, and any process can be interrupted and moved out of execution by the kernel. This is known as thread management.
- Files are stored on disk in a hierarchical file system, with a single top location throughout the system (root, or "/"), with both files and directories, subdirectories, sub-subdirectories, and so on below it.
- With few exceptions, devices and some types of communications between processes are managed and visible as files or pseudo-files within the file system hierarchy. This is known as *everything is a file*. However, Linus Torvalds states that this is inaccurate and may be better rephrased as "everything is a stream of bytes".

The UNIX operating system supports the following features and capabilities:

- Multitasking and multiuser
- Programming interface
- Use of files as abstractions of devices and other objects
- Built-in networking (TCP/IP is standard)
- Persistent system service processes called "daemons" and managed by init or inet

(b) How do you add, modify and delete a user in Unix environment? [03]

### Adding a User

we use the useradd command to add a new user to the UNIX system.

```
# useradd -u 4000 -md /home/someuser -c "Some User" -
g people -G specialpeople,others -s /bin/bash
someuser

# useradd -u uid -md homedir -c "comment" -g group -G
additional groups -s usershell username
```

### Modifying a User with usermod

We use the usermod command to modify the attributes of the user

```
# usermod -aG folks -s /bin/ksh someuser
```

### Deleting a User

```
# userdel -r someuser
```

The `-r` flag in the above command tells userdel to also remove the user's home directory which is not the default behavior.

2 (a) Explain "man" documentation and its internal command.[07]

The *man* command is used to format and display the *man pages*.

Its basic syntax is

```
man [option(s)] keyword(s)
```

For example, the following provides information about the *ls* command

```
man ls
```

As another example, the following displays the man page about the man pages:

```
man man
```

man automatically sends its output through a *pager*, usually the program *less*. A pager is a program that causes the output of any program to be displayed one screenful at a time, rather than having a large amount of text scroll down the screen at high (and generally unreadable) speed.

less writes a colon at the bottom of the screen to indicate the end of the on-screen page. The user can move to the next page by pushing the space bar and can return to the previous page by pressing the *b* key. Pressing the *q* exits the man pages and returns the user to the shell program.

Each man page is a self-contained article that is divided into a number of sections, the headers for which are labeled with upper case letters. The sections for commands are typically something like NAME, SYNOPSIS, DESCRIPTION, OPTIONS, AUTHOR, BUGS, COPYRIGHT, HISTORY and SEE ALSO, although there may be some differences according to the particular command.

Also, the man pages as a whole are organized into sections, each containing pages about a specific category of topics as shown below. The section to which an article belongs is indicted in parenthesis in the top line, before the NAME header.

1. executable programs or shell commands
2. system calls
3. library routines
4. special files (i.e., *devices* in the */dev* directory)
5. file formats
6. games
7. macro packages
8. system administration commands
9. kernel routines

n. Tcl/Tk (a programming language)

The syntax to specify an article from a particular section is:

```
man section_number keyword
```

Thus, for example, the following would display the article about mount from Section 2 instead of from the default Section 8:

```
man 2 mount
```

(b) Explain Shell variables : a) SHELL b) HOME c) PATH [03]

**SHELL** – is the environment variable that is set during login, to be the pathname of the Unix Shell you are assigned in the Unix password file.

$echo $SHELL

/bin/bash

**HOME** – the home directory of the current user. It is set during login to be the absolute path to your HOME directory as set in the Unix password file.
$echo $HOME
/home/student

**PATH** – shell command search path. PATH is set during login to be a colon-separated list of directories in which the shell will look when it tries to find an executable file that matches a command name.

3 (a) Explain briefly the file attributes listed using **ls -l** command.[06]

The very first character identifies the file type. A dash ( – ) represents a normal file while a “d” represents a directory.

```
ls -l /path/to/directory
total 128
drwxr-xr-x 2 archie users  4096 Jul  5 21:03 Desktop
drwxr-xr-x 6 archie users  4096 Jul  5 17:37
Documents
```

`Type:`

Whether ordinary, directory, device, etc.

`Permissions:`

Determines who can read, write or execute a file.

- The next 3 characters represent the owner permissions. Each letter represents the presence of a permission and a dash ( – ) represents the absence of a permission. In the above example, the owner has read and write permissions.
- The next 3 characters represent the group permission. So, in the above example, you can see that the “group” has the ability to read but not write or execute.

- Also, finally the last 3 characters represent the permissions for others. In the above example, the file has only execute permission for “others”.

`Links:`

Number of hard links to the file. A number of files in the file system can actully reference the same file on the drive.

`Owner:`

A file is owned by a user, by default its creator. The owner can change many file attributes and set the permissions.

`Group Owner:`

The group which owns the file. The owner by default belongs to this group.

`File Size:`

Number of bytes of data contained.

`File Time Stamps:`

- Date and time of last modification
- Date and time of last access

The last field represents the file name.

(b) What are internal and external commands? Explain **type** command with examples. [04]

Some commands that you type are *internal*, which means they are built into the shell, and it's the shell that performs the action. For example, the *cd* command is built-in.

The *ls* command, on the other hand, is an *external* program stored in the file */bin/ls*.

The shell doesn't start a separate process to run internal commands. External commands require the shell to *fork* and **exec** a new **subprocess.**

4 (a) Explain the below mentioned commands with its usage in detail,

a) cal b) date c) echo d) who
e) bc f) printf [10]

a)cal
Displays current month calendar on console:

```
cal 11 2012
```

```
Displays selected month and year calendar on console:
```

```
b)date
```

**date** command is used to display the system date and time. date command is also used to set date and time of the system. By default the date command displays the date in the time zone on which unix/linux operating system is configured.You must be the super-user (root) to change the date and time.

**Syntax:**

```
date [OPTION]... [+FORMAT]
```

**List of Format specifiers used with date command:**

**%D: Display date as mm/dd/yy.**
%d: Display the day of the month (01 to 31).
%a: Displays the abbreviated name for weekdays (Sun to Sat).
%A: Displays full weekdays (Sunday to Saturday).
%h: Displays abbreviated month name (Jan to Dec).
%b: Displays abbreviated month name (Jan to Dec).
%B: Displays full month name(January to December).
%m: Displays the month of year (01 to 12).
%y: Displays last two digits of the year(00 to 99).
%Y: Display four-digit year.
%T: Display the time in 24 hour format as HH:MM:SS.
%H: Display the hour.
%M: Display the minute.
%S: Display the seconds.

**Syntax:**

**$date +%[format-option]**

c) echo

The echo command tells the computer what to do, while the argument is a command’s input data. By default, standard output is the display screen but it can also be redirected to a printer or a file. For echo, the syntax is:

```
echo [option(s)] [string(s)]
```

```
echo -e "\n Projects: \n\n\tplan \n\tcode \n\ttest\n"
```

d)who

The standard Unix command `who` displays a list of users who are currently logged into the computer.

```
$ who
hduser   tty7         2018-03-18 19:08 (:0)

$ who -T -H
NAME       LINE         TIME             COMMENT
hduser   + tty7         2018-03-18 19:08 (:0)
```

e)bc

**bc** command is used for command line calculator. It is similar to basic calculator by using which we can do basic mathematical calculations. Arithmetic operations are the most basic in any kind of programming language. Linux or Unix operating system provides the **bc command** and **expr command** for doing arithmetic calculations. You can use these commands in bash or shell script also for evaluating arithmetic expressions.

f)printf

**printf** prints a *formatted string* to the standard output.

```
printf "My name is \"%s\".\nIt's a pleasure to meet
you." "John"
```

This command produces the output:

```
My name is "John". It's a pleasure to meet you.
```

5 (a) Assuming that a file current permissions are **rw-r-xr--**, specify the **chmod** expression both in relative and absolute methods to achieve the following,
i) rwxrwxrwx ii) r--r----- iii) ---r--r-- iv)---------
v) r---w---x [06]

Relative mode:

i) chmod u+x,g+w,o+wx

ii) chmod u-w,g-x,o-r filename
iii) chmod u-rw,g-x filename
iv) chmod u-rw,g-x,o-r filename
v) chmod u-w,g-rx,g+w filename

Absolute mode:

i) chmod 777 filename
ii) chmod 440 filename
iii) chmod 044 filename
iv) chmod 000 filename
v) chmod 421 filename

(b)Explain the significance of the following special characters in UNIX [4]
i) **.** (Single dot) ii) **..** (double dot) iii) **~** (tilde)

i) . (single dot):
Single dot represents the current directory in the path name.

ii) .. (double dot)
The double dot represents the parent directory in the path name.

Iii) ~ (tilde)
~ (tilde) represents the home directory for example for user1 the home directory is /home/user1 then ~ represents /home/user1.

~user2 represents /home/user2

6 Explain the below mentioned commands with its usage in details[10]
a) cat b) cp c) rm d) mv e) wc f) od

**a)cat**
The cat command concatenate files and displays the contents of files on the standard output (terminal).
Cat command accepts more than one filename as agruments.
Syntax:
cat [OPTION]... [FILE]...

Numbering lines (-n) option:
The -n option numbers lines.

Example:
$ cat -n file2
1 01|accounts|6213
2 02|progs|5423
3 03|marketing|6521
4 04|personnel|2356

**b)cp**
cp command copies a file or a group of files.
It creates a exact image of the file on disk with a different name.
If the destination file doesn’t exist, it will first be created before copying.

If the destination file is already present it will be overwritten without any warning from the system.

If there is only one file to be copied, the destination can be either an ordinary file or directory.

If we want to copy more than one file with a single invocation of cp command, then the destination must be a directory.

SYNTAX:

cp [OPTION]... SOURCE DEST <---- Copies SOURCE file to DEST file

cp [OPTION]... SOURCE... DIRECTORY <--- copies multiple SOURCES to DIRECTORY

EXAMPLES:

$ cp file1 newfile1 <--- contents of file1 is copied to a new file – newfile1

<--- If newfile1 is not present it will be created before copying

<--- If newfile1 is already present it will be overwritten

$ cp file1 USP/newfile1 <--- copies file1 to newfile1 under USP directory

$ cp file1 USP <--- copies file1 with the same name under USP directory

$ cp USP/file1 . <--- copies USP/file1 to current directory

$ cp file1 file2 file3 USP

**c) rm**

rm stands for remove. The rm command removes files or directories

SYNTAX:

rm [OPTION]... [FILE]...

The file once deleted cannot be recovered.

Examples:

$ rm newfile new <----- deletes two files newfile and new

$ rm * <----- deletes all files in the current directory

OPTIONS OF rm COMMAND:

Interactive deletion (-i):

The -i option makes the rm command ask the user for confirmation before removing each file.

Example:

$ rm -i prog6.c prog7.c

rm: remove regular file 'prog6.c'? n

rm: remove regular file 'prog7.c'? [Enter] <--- No response – file not

deleted
Recursive Deletion (-r or -R):
With the -r (or -R) option, rm performs a tree walk – a thorough recursive search for all subdirectories and files within these subdirectories. At each stage, it deletes everything it finds.
The command rm won't remove directories, but when used with -r or -R option, even directories are removed.
$ rm -r * <----- will delete all files in the current directory and all its subdirectories. If there is no backup, then these files will be lost forever.

Forcing Removal (-f) option:
The commad rm prompts for removal if a file is write-protected. The -f option overrides this protection and forces removal.
$ rm -rf * <------ Most Dangerous commad.
Deletes everything in the current direcotry and below
NOTE: Be doubly sure before you use rm -rf * as it removes everything. If the root user invokes rm -rf * in the / directory the entire UNIX system will be wiped out from the hard disk.

**d)mv**
The mv command moves (renames) files. It has two distinct functions :
1. It renames a file (or directory)
2. It moves a group of files to a different directory.

The mv command doesn't create a copy of the file, it just renames it. No additional space is consumed on disk during renaming.
If the destination file doesn't exist, it will be ceated.
By default, mv command doesn't prompt for overwriting the destination file it it exists.
The -i (interactive) option will prompt before overwriting.
mv command can also be used to rename a directory.

Syntax:
mv [OPTION]... [-T] SOURCE DEST
mv [OPTION]... SOURCE... DIRECTORY
-T option – treats DEST as normal file.

Examples:
1. $mv file3 newfile <----- file3 is renamed as newfile
2. $mv file1 file2 Desktop <------ set of files – file1 file2 are moved to Desktop directory.
3. $mkdir temp
$ mv temp temporary <-------- mv command used to rename a directory

**e)wc**

The command wc counts lines, words and characters in one or more files.
SYNTAX:
wc [OPTION]... [FILE]...
EXAMPLES:
$ cat rahul.c
#include <stdio.h>
int main()
{
printf("301");
return 0;
}
$ wc rahul.c
6 9 59 rahul.c <----- wc displays four column output
The wc command displays a four-columnar output. The four columns are number of lines, number of words, number of characters and filename.
A line is any group of characters not containing a newline.
A word is a group of characters not containing a space, tab or newline. A character is the smallest unit of information, and includes a space, tab and
newline.
Lets display the contents of files input.txt and rahul.c

**f)od**

od stands for octal dump.
od command displays or dumps the contents of a file in various formats. As it displays the data in octal format by default, the command is given the name od (octal dump).
Syntax:

od [OPTION]... [FILE]...
Let’s create a file input.txt
$ cat > input.txt
100
101
102
103
104
105
-b option: The -b option displays the contents of the file in octal format.
$ od -b input.txt
0000000 061 060 060 012 061 060 061 012 061 060 062 012 061 060 063 012
0000020 061 060 064 012 061 060 065 012
0000030
The first column in the output of od represents the byte offset in file.

7(a) Write significance of the following commands [3]
i) date ; who > myfile ii) ls -ld /

i) date ; who > myfile

In the above command the two commands date and who are executed one by one and the output of date command is displayed onto the monitor, but the output of who command is redirected to the file – myfile.

ii) ls -ld /

output:

drwxr-xr-x 23 root root 4096 Oct 4 16:17 /

The above command gives the long list information of the directory root. The output is one line with all details of the root directory.

(b) How do you set terminal characteristics? Explain with examples.[7]

**stty command** changes and prints terminal line settings.
**stty** displays or changes the characteristics of the terminal.

`stty sane`
Reset all terminal settings to "sane" values; this has the effect of "fixing" the terminal when another program alters the terminal settings to an unusable condition.

`stty -echo`
Disable echoing of terminal input.

`stty echo`
Re-enable echoing of terminal input.

`stty -a`
Display all current terminal settings.

8(a) Explain the below mentioned commands with its usage: [7]
i) pwd ii) cd iii) mkdir iv) rmdir

**i) pwd**
Command: pwd
The command pwd stands for print working directory.
As the name says, prints the working or current directory.
The command pwd displays the absolute pathname of working directory.
Its an internal command.
Syntax:
$ pwd
/home/student

**ii) cd**
Command cd
The command cd stands for change directory. We can move around in the file system by using the cd command.
cd changes the working directory. It is an internal command.
cd without any arguments, changes to home directory of the user i.e., the directory where the user originally logged into. So, to move directly to the home directory, use cd command without any arguments.

When used with an argument, cd changes the current directory to the directory specified as argument.

The cd command can sometimes fail if you don’t have proper permissions to access the directory.

Syntax:
cd [arguments]

**iii) mkdir**

The mkdir command creates a directory.

mkidr stands for make directory.

Syntax:

$mkdir directory_name...

Examples:

1) $mkdir USP <- The directory USP is created under the current directory.

2) $mkdir co ade <- Two directories co and ade are created under current directory.

3) $ mkdir dslab/ds ds

mkdir: cannot create directory ‘dslab/ds’: No such file or directory

In the above example, we created two directories – dslab/ds and ds. The order of specifying the arguments is important. We cannot create a subdirectory before creation of its parent directory. Therefore creation of dslab/ds directory failed, but cretion of ds directory will be successful.

4) Sometimes, mkdir fails to create a directory.

$ mkdir test

**iv) rmdir**

The rmdir (remove directory) command removes directories.

Syntax:

rmdir directory_name...

The rmdir command cannot delete a directory unless it is empty. If it contains any files, UNIX will return an error message, “Directory not empty.”

You can’t remove a subdirectory unless you are placed in a directory which is

hierarchically above the one you have chosen to remove, i.e., you cannot remove a directory being in that directory.

Examples:

1) rmdir co ade ds

NOTE:

1) The mkdir and rmdir commands work only in directories owned by the user. User can create and remove subdirectories in his/her home directory or any subdirectories created by the user.

2) User will not be able to create or remove files and directories in other users' directories.

(b)Write the output of the following commands: [3]

i) cal 8 1947 ii) echo "today's date is `date`"

iii) date +"Date is :%a/%h/%Y"

i) displays calendar for the month of August of year 1947

ii) today's date is Thu Oct 11 15:36:03 IST 2018

iii) Date is : Fri/Sep/2018