PADMASHREE KRUTARTHA ACHARYA INSTITUTE OF ENGINEERING & TECHNOLOGY BARGARH

Operating system lab manual

4th semester

Computer science & engineering

Pr.1-OPERATING SYSTEM LAB

Total Periods	60	Maximum Marks	50 Marks
Lab. Periods:	4 Periods /week	Term Works	25 Marks
Examination	3hours	End Semester Examination	25Marks

A. LIST OF PRACTICALS:-

1. Write a Shell script to print the command line arguments in reverse order.
2. Write a Shell script to check whether the given number is palindrome or not.
3. Write a Shell script to sort the given array elements in ascending order using bubble sort.
4. Write a Shell script to perform sequential search on a given array elements.
5. Write a Shell script to perform binary search on a given array elements.
6. Write a Shell script to accept any two file names and check their file permissions.
7. Write a Shell script to read a path name, create each element in that path e.g: a/b/c i.e., 'a' is directory in the current working directory, under 'a' create 'b', under 'b' create 'c'.
8. Write a Shell script to illustrate the case-statement.
9. Write a Shell script to accept the file name as arguments and create another shell script, which recreates these files with its original contents.
10. Write a Shell script to demonstrate Terminal locking.
11. Write a Shell script to accept the valid login name, if the login name is valid then print its home directory else an appropriate message.
12. Write a Shell script to read a file name and change the existing file permissions.
13. Write a Shell script to print current month calendar and to replace the current day number by '*' or '**' respectively.
14. Write a Shell Script to display a menu consisting of options to display disk space, the current users logged in, total memory usage, etc. (using functions.)
15. Write a C-program to fork a child process and execute the given Linux commands.
16. Write a C-program to fork a child process, print owner process ID and its parent process ID.
17. Write a C-program to prompt the user for the name of the environment variable, check its validity and print an appropriate message.
18. Write a C-program to READ details of N students such as student name, reg number, semester and age. Find the eldest of them and display his details.

Books Recommended:-

Sl.No	Name of Authors	Title of the Book	Name of the publisher
1	Sumitabha Das, 4th Edition,	"UNIX – Concepts and Applications",	Tata McGraw Hill, 2006.
3	Yashvant Kanetkar	Unix Sell Programming 1st edition	BPB Publication

DOS COMMANDS

Introduction   Some course material on the Teaching Network may still use the command line operating system called DOS (Disk Operating System). This requires the user to type specific commands at a prompt.

You may also need to use this system, for example, when changing your password, and you can enter DOS by choosing 'MS-DOS Prompt' from the Public menu on the taskbar.

Commands can be typed in response to the network DOS prompt:

Command Format

       G:\>

Commands may be typed in upper or lower case. Here, they are specified in upper case to distinguish them from other input.

Commands have a particular format. Some can be used simply by typing the command name only, as in:

       DIR

which displays a listing of files in the current directory.

Many commands can take extra information called parameters to make them more specific, For example, the COPY command can have a parameter specifying a file to be copied and a parameter specifying a file to which the copy is to be made. So the command would look like this:

Parameters

       COPY *filea fileb*

In this case, *filea* would be the name of a file to copy from and *fileb* would be the name of a file to copy to.

Parameters can sometimes be omitted in which case a default action will apply.

Commands can also be qualified by the use of options. Options are preceded by a / sign. A simple example of the use of an option is with the DIR command. DIR can be qualified by /P or /W. Thus the commands:

Options

       DIR
       DIR /W
       DIR /P

result in listings presented in a different format as detailed later.

Some Useful DOS Commands

COPY   Make a copy of a file or merge files together.

> COPY *original-file destination-file*

where *original-file* and *destination-file* are file names, separated by a space.

The command can be used to merge several files into one file since the *original-file* parameter can be a list of filenames separated by the + symbol. For example:

> COPY file1 + file2 file3

would copy file1 and file2 into file3.

Note that the file names used in the copy command *m ust* include the file extension if it exists. So if you were copying a fortran program called first.for, for example, you would need to include the .for extension in the filename.

DEL   Delete a file.

> DEL *filename*

where *filename* is the name of the file to be deleted. You are asked to confirm that you really want to delete the file. Note that the name must include the file extension.

The amount of space on a disk is limited and it is necessary periodically to tidy up unwanted files. It is possible to use a '*' character in a filename to match any sequence of characters. This feature is very useful when deleting unwanted files with identical extensions. For example, when developing programs,

> DEL *.obj

will delete every binary (.obj) file in the current directory. Since an .obj file can always be recreated from the original source program, it is usually unnecessary to keep it.

DIR   Obtain a list of the files stored in a directory.

If used without options this command will give a list of files in the current directory, including any extension (e.g. .for), and their size.
If used with the option /P, that is:

> DIR /P

The same information will be displayed page by page with the message:

> strike a key when ready

at the end of each page.
If it is used with the /W option, i.e.

> DIR /W

the listing is of names and extensions only and in a more compact format across the page.

DISKCOPY   Take a security copy of your working disk.

It is very important to have a second copy of your programs and data in case you lose your disk or it becomes unreadable for some reason. Use the DISKCOPY command as follows:

DISKCOPY A: A:

where A: denotes the drive the disks will be loaded into. When a copy of your disk is generated in the computer's memory, the computer tells you to remove the first or source disk and insert the disk, which is to become the copy (the target disk).

Insert the target disk and press ENTER. If the second disk is not formatted it will be formatted automatically. The source disk should be write protected using the tab in the corner in case you mistake it for the target disk (see the chapter on using disks).

Obtain a printed listing of a file.

LIST
LIST *filename*

where *filename* is the file to be printed. If the printer is busy, or if you don't need a printed copy, you can use the TYPE command to view a file on the screen.

You could also use an editor such as DOS Edit to display a file, in which case, as a bonus, you would then be in a position to correct any errors you might notice. Remember that the file name must include the file extension. So if you want to print your Pascal program called first you would need to use:

LIST first.pas

Change the name of a file. This command can be abbreviated to REN:

RENAME

REN *oldname newname*

where *oldname* is the file to be renamed and *newname* is the name it is to be given.

Note that files on drives other than the current drive can be referred to by prefixing the filename by the relevant drive letter, for example:

A:*filename*

where the \ refers to the 'root' directory of the drive.
Filenames can also include directories separated by the \ character as in:

A:*dir1**dir2**filename*

TYPE   View a file on the screen.

TYPE *filename*

where filename is the name of the file to be displayed on the monitor screen. If you are looking at a long file, you will need to press the PAUSE key to stop continuous scrolling. Press the space bar to continue scrolling when you have read the screen.

To format a floppy disk, place the disk in the floppy drive and type:

FORMAT

FORMAT A:

Note that by default the disk will be formatted to hold about 1.44 megabytes of data.

Warning: formatting will destroy any data on the disk so only do this once!

DOS Command Listing

In the following section, alternative options are separated by the '!' character. Arguments are optional unless in italics. 'cwd' stands for 'Current Working Directory'.

DOS commands are not case sensitive. Some commands have switches; these must be preceded by a forward slash (/).

DOS Conventions

Pathnames may be preceded by a drive letter as in
'X:pathname' and if no drive or pathname is given the current directory is assumed.

ATTRIB +!-r +!-a *pathnam e*
          display, or set!clear Read-only or

The Commands   Archive attribute

BREAK on!off

display, or turn on!off increased level of ^C detection

CHDIR (CD) path display, or change working directory

CHKDSK pathname (A: drive only) check disk or file logical
          structure /f - fixproblems encountered
          /v   - verbose; displays filespecs

CLS
          clear screen
COMMAND path cttydev
          run nested CLI from path with I/O device cttydev
          /e:#   - set environment size #
          /p   - do not invoke another CLI
          /c command - run command and then enter

COPY pathname pathname
or

COPY *pathnam e + pathnam e*
> copy or concatenate files
> /v   - verify writes
> /a!b   - preceding and all subsequent files are ascii! binary (* filenames in source(s) and target are matched one-to-one.)

CTTY device change MS-DOS I/O device

DATE dd-mm-yy
> display and/or set date (numerical country-dependent format)

DEL ! ERASE *pathnam e*
> delete file(s) - prompts if pathname is *.*

DIR pathname
> display directory - filename and/or ext default to *
> /p   - paginate
> /w   - multi-column

DISKCOPY drive: drive:
> copy disk sector-by-sector rather than file-by-file

EXIT
> exit nested CLI

FIND "*string*" pathname find and display lines containing string in file
> /v   - lines NOT containing string
> /c   - count lines only
> /n   - display line numbers also

FORMAT *drive*: (A: drive only)
> /1   - single sided
> /4   - use double rather than high density (40 tracks)
> /8   - use 8 sectors of each track
> /n:xx - specifies xx sectors per track
> /t:yy   - specifies yy tracks
> /v   - prompt for volume label, up to 11 characters
> /f:720 - format at 720 kb

LABEL drive: label (A: drive only) display, or edit volume label - 11 characters
> excluding most specials

MKDIR (MD) path make directory
MODE
> interactively configure various options

MORE
> paginates screen output
> e.g. type filename | more

PATH path;path;... display or set command search path(s) to be used after cwd

PATH;
> resets default to cwd only

PROMPT string
> reset or set prompt.

Characters (each prefixed by $) mean: $=$, t=time, d=date, p=cwd, v=version, n=drive, g=>, l=<, b=!, _=CRLF, s=leading space, e=ESC (for ANSI driver)

RENAME (REN)*pathnam e pathnam e*
    rename file(s) within a drive - wildcards are matched one-to-one

RMDIR (RD) path remove empty directory

SET variable=text display all, set or clear MS-DOS variable - accessed as
    %variable%

SHARE
    enable networked multi-access file locking
    /f:nbytes - nbytes per file, need about 20 bytes per file,
    default 2048
    /l:nlocks - nlocks per file, default 20

SORT pathname pathname
    sort lines of file alphabetically, ignoring case, to file or
    stdout; reads stdin by default
    /r   - reverse order
    /+n   - on n'th character in each line, default is first

SUBST drive: path (A: drive only) display substitutions, or substitute path by virtual
    drive

SYS *drive*: (A: drive only) copy hidden MS-DOS .sys files from default drive

TIME hours:minutes display and/or set time using 24-hour format

TYPE *pathnam e*
    output contents of a file, with tab spacing of 8

VER
    display MS-DOS version
VERIFY on!off
    display, or set!clear disk write verification

VOL drive:
            display disk volume label

XCOPY *pathnam e* pathname
            copy directory tree
    /a!m   - if archive bit set ! also clears in source
            /d:date- modified on or after date only
    /s!e   - copy subdirectories if not empty ! even if empty
    /p   - prompt
    /v   - verify
    /w   - wait for keypress

If renamed to MCOPY it determines automatically whether target is file or directory.

On-Line Command Help	Full details of all DOS commands can be seen by following the command with the switch /? For example,

DIR /?

gives a listing of all DIR options.

Command Editing	DOS commands are stored in a template and previous commands can be recalled, character by character enabling editing as required.

The template is accessed by the following keys:-

F1   -   get next character from template
F2 C   -   get characters up to but excluding character C F3
            -         get remaining characters from template del   -
        skip one character in template
F4 C   -   skip characters up to but excluding character C
ESC   -   clear command line
INS   -   toggle overwriting of template
F5 - copy command to template for re-editing F6 - put ^Z in new template

The arrow keys may also be used to recall the previous command.

Batch File Commands	All DOS commands may be used in batch files. Additionally, the following commands are useful for more advanced batch processes.

Arguments for batch files are accessed as '%1' to '%9'.

ECHO on!off!message display echo status, turn echoing on!off (default on) or
        display message

    FOR %%C IN (SET of items) DO command
    C is any character other than 0-9, SET is e.g. list of files
GOTO LABEL
        LABEL is any line of text, usually preceded by colon (:) in first column, which
        makes MS-DOS ignore it other than as a label. Terminates if label not found

    IF ERRORLEVEL *num ber com m and*
    command executed if previous command returned exit code >= number

IF *string1 == string2* command command executed if strings match
        may be negated by NOT before condition

IF EXIST *filenam e com m and* command executed if file exists may
        be negated by NOT before condition

PAUSE comment
            comment displayed only if echo on

REM comment
            remark - ignored by MS-DOS

SHIFT
        shift arguments - allows access to more than 9

Special Characters	Several special characters may be used when referring to directories and files:

\   =   root directory or a directory separator
.   =   current directory ..
         =   parent directory

Wildcard characters may be used in filenames or extensions:

?   =   any character
*   =   any tail or extension
X: - switches to current working directory (cwd) on drive X.

File Comparison Utility	The FC command enables comparison of two files:

FC *pathnam e1 pathname2*

      compare two files, or two wildcarded sets of files
      /a   - abbreviate output of ASCII comparison
      /b   - force binary comparison (byte-by-byte)
      /c   - ignore case
      /L   - force ASCII comparison (line-by-line)
      /Lb#   - use line buffer of # lines
      /n   - display line numbers in ASCII mode
      /t   - do not expand tabs - default expands to spacing
         of 8
      /w   - compress white space (tabs and spaces) to single space (leading or trailing white space always ignored)
      /# - # lines must match to re-synchronize else regarded different (default is 2)

I/O Control
DOS input and output may be controlled by the following control key sequences. (^ = the CTRL key).

^C   -   abort current command
^H   -   destructive backspace
^J   -   linefeed - physical newline to input long lines
^N   -   toggle copying of terminal output to printer
^P   -   toggle redirection of terminal output to printer
^S   -   suspend/restart terminal output
^X   -   cancel current line, and output \ -CR-LF
^Z   -   end of file

| I/O Redirection | Input and output from commands or programs can be redirected by using the following symbols: |

>   send output to...
>>   append output to...
<   take input from...
| pipe output to next input

For example:
pipe output from command a to input of command b:
      command a | command b

send a directory listing to file filename: DIR >
      filename

| Using The DOS Editor | DOS includes a full-screen editor invoked by the EDIT command (with or without a filename). To use this editor type:
      EDIT (*filenam e*)

The editor provides pull-down menus, operated by the keyboard or mouse, and a help facility. Note that any changes made to a file overwrites the original, no backup is created. |

COMMANDS IN LINUX

1. alias: The alias command lets you give your own name to a command or sequence of commands. You can then type your short name, and the shell will execute the command or sequence of commands for you.

alias cls=clear

This sets up an alias called cls for clear command so whenever we use cls command it will clear the screen as done by the clear command.

alias pf="ps -e | grep $1"

This alias uses the ps command to list the running processes and then pipes them through the grep command. The grep command looks for entries in the output from ps that match the command line parameter $1.)

If you wanted to discover the process ID (PID) of the shutter process—or to find out if shutter was even running—you could use the alias like this. Type pf, a space, and the name of the process you are interested in:

pf shutter

2. cat: The cat command (short for "concatenate") lists the contents of files to the terminal window. To read the contents of your .bash_log_out file,

cat .bash_logout)

if the content doesnot fit in a single window the use | with "less" command to see page by page. Type "q" to exit from 'less'.

cat .bashrc | less

3. cd: The cd command changes your current directory. If you are changing to a directory that is within your current directory, you can simply type cd and the name of the other directory.

cd work

If you are changing to a directory elsewhere within the filesystem directory tree, provide the path to the directory with a leading /.

```
cd /usr/local/bin
```

To quickly return to your home directory, use the ~ (tilde) character as the directory name.

```
cd ~
```

You can use the double dot symbol .. to represent the parent of the current directory. You can type the following command to go up a directory:

```
cd ..
```

Imagine you are in a directory. The parent directory has other directories in it, as well as the directory you're currently in. To change into one of those other directories, you can use the .. symbol to shorten what you have to type.

```
cd ../games
```

4. **chmod:** The chmod command <u>sets the file permissions flags</u> on a file or folder. The flags define who can read, write to or execute the file.

```
ls –l myshell.sh
-rwxrwxrwx. . . . . . . . . . . . . . . .myshell.sh
```

- represent a file
d represents a directory
1st rwx represents file permission of owner
2nd rwx represents file permission of group
3rd rwx represents file permission of other
If it shows rwx then read,write and execute permission is allowed and if it shows "–" then that particular permission is not granted.
"chmod" has following file permission parameters.
- **0:** No permission
- **1:** Execute permission
- **2:** Write permission
- **3:** Write and execute permissions
- **4:** Read permission
- **5:** Read and execute permissions
- **6:** Read and write permissions

- **7: Read, write and execute permissions**

To set the permission to be read, write, and execute (7 from our list) for the *owner*; read and write (6 from our list) for the *group* ; and read and execute (5 from our list) for the *others* for the file example.txt we'd need to use the digits 765 with the chmod command :

chmod -R 765 example.txt

5. **chown:** The chown command allows you to change the owner and group owner of a file.

ls –l myshell.sh
-rwxrwxrwx -1 dave dave 5957 Feb 10 15:58 myshell.sh

The first of these indicates the name of the file owner, which in this case is the user dave. The second entry shows that the name of the group owner is also dave. we can use chown to change the owner or group, or both of a file with "sudo" command.

sudo chown dave:mary myshell.sh

it will change the group owner to mary from dave but owner name will remain intact. To change both the owner and the group owner to mary, you would use the following command;

sudo chown mary:mary myshell.sh
ls –l myshell.sh
-rwxrwxrwx -1 mary mary 5957 Feb 10 15:58 myshell.sh

6. **df:** The df command <u>shows the size, used space, and available space</u> on the mounted filesystems of your computer. Two of the most useful options are the -h (human readable) and -x (exclude) options. The human-readable option displays the sizes in Mb or Gb instead of in bytes. The exclude option allows you to tell df to discount filesystems you are not interested in. For example, the squashfs pseudo-filesystems that are created when you install an application with the snap command.

df -h -x squashfs

7. **diff:** The diff command <u>compares two text files</u> and shows the differences between them. The -y (side by side) option shows the line differences side by side. The -w (width) option lets you specify the maximum line width to use to avoid wraparound lines. Let the two files are called alpha1.txt and alpha2. The --suppress-common-lines prevents diff from listing the matching lines, letting you focus on the lines which have differences.

diff -y -W 70 alpha1.txt alpha2.txt

The --suppress-common-lines prevents diff from listing the matching lines, letting you focus on the lines which have differences.

```
diff -y -W 70 alpha1.txt alpha2.txt --suppress-common-lines
```

8. **echo:** The echo command prints (echoes) a string of text to the terminal window.

```
echo "hello how are you"
```

The echo command can show the value of environment variables, for example, the $USER, $HOME, and $PATH environment variables. These hold the values of the name of the user, the user's home directory, and the path searched for matching commands when the user types something on the command line.

```
echo $USER
echo $HOME
echo $PATH
```

9. **exit:** The exit command will close a terminal window, end the execution of a shell script, or log you out of an SSH remote access session.

```
exit
```

10. **whoami:** Use whoami to find out who you are logged in as or who is logged into an unmanned Linux terminal.

```
whoami
```

11. **w:** The w command lists the currently logged in users.

```
w
```

12. **uname:** You can obtain some system information regarding the Linux computer you're working on with the uname command.
- Use the -a (all) option to see everything.
- Use the -s (kernel name) option to see the type of kernel.
- Use the -r (kernel release) option to see the kernel release.
- Use the -v (kernel version) option to see the kernel version.

```
uname -a
uname -s
uname -r
uname -v
```

13. top: The top command shows you a real-time display of the data relating to your Linux machine. The top of the screen is a status summary.

The first line shows you the time and how long your computer has been running for, how many users are logged into it, and what the load average has been over the past one, five, and fifteen minutes.

The second line shows the number of tasks and their states: running, stopped, sleeping and zombie.
- D: Uninterruptible sleep
- R: Running
- S: Sleeping
- T: Traced (stopped)
- Z: Zombie

The third line shows CPU information. Here's what the fields mean:
- us: value is the CPU time the CPU spends executing processes for users, in "user space"
- sy: value is the CPU time spent on running system "kernel space" processes
- ni: value is the CPU time spent on executing processes with a manually set nice value
- id: is the amount of CPU idle time
- wa: value is the time the CPU spends waiting for I/O to complete
- hi: The CPU time spent servicing hardware interrupts
- si: The CPU time spent servicing software interrupts
- st: The CPU time lost due to running virtual machines ("steal time")

The fourth line shows the total amount of physical memory, and how much is free, used and buffered or cached.

The fifth line shows the total amount of swap memory, and how much is free, used and available.

The user has pressed the E key to change the display into more humanly digestible figures instead of long integers representing bytes. Press the Q key to exit from top.

top

tar: With the tar command, we can create an archive file (also called a tarball) that can contain many other files. we can also use tar to extract the files from an archive file.

tar [options] [archive-file] [file or directory to be archived]
Options:
-c : Creates Archive
-x : Extract the archive
-f : creates archive with given filename
-t : displays or lists files in archived file
-u : archives and adds to an existing archive file
-v : Displays Verbose Information
-A : Concatenates the archive files
-z : zip, tells tar command that create tar file using gzip

-j : filter archive tar file using tbzip

-W : Verify a archive file

-r : update or add file or directory in already existed .tar file

$ tar cvf file.tar *.c

Output: os2.c

os3.c

os4.c

It will create a tar file file.tar which is the archive of all .c files in the current directory

Extracting files : This command extracts files from Archives.
$ tar xvf file.tar

Output :
os2.c

os3.c

4.c

zip the Archive (-z): This command creates a tar file called file.tar.gz which is the Archive of .c files.
$ tar cvzf file.tar.gz *.c

Extracting a zip Archive : This command extracts files from tar archived file.tar.gz files.
$ tar xvzf file.tar.gz

15. **sudo:** this command required when performing actions that require root or superuser permission such as changing the password of another user.

sudo passwd mary

16. **shutdown:** this command shutdown and reboot the linux system.Using shutdown with no parameters will shut down your computer in one minute.
shutdown

To shut down immediately, use the now parameter.

shutdown now

17. **pwd:** This command refers working directory from the root/directory.

pwd

18. **ps:** this command lists running processes.

ps [option]

blank space: list the processes running in the current shell)

-u: list the process running to a particular user.

-e: to see every running process.

```
Ps
OR
ps -u dave | less
OR
ps -e | less
```

19. **ping:** The ping command lets you verify that you have network connectivity with another network device. It is commonly used to help troubleshoot networking issues. To use ping, provide the IP address or machine name of the other device. The ping command will run until you stop it with Ctrl+C.

```
ping 192.168.4.18 )
```

To ask ping to run for a specific number of ping attempts, use the -c (count) option.

```
ping -c 5 192.168.4.18
```

To hear a ping, use the -a (audible) option.

```
ping -a 192.168.4.18
```

20. **mv:** The mv command allows you to move files and directories from directory to directory. It also allows you to rename files.

mv [file name to be moved] [destination directory].

To move apache.pdf from the "~/Document/Ukulele" directory to current directory, (.) we can write

```
mv ~/Documents/Ukulele/Apache.pdf . )
```

To rename:

```
mv ~/Documents/Ukulele/Apache.pdf ./The_Shadows_Apache.pdf
```

21. **mkdir:** The mkdir command will create new directories in the filesystem. If the new directory is in the current directory then path is not required otherwise the path to the new directory is required.

The following commands create two new directories in the current directory called "invoices" and "quotes,"

mkdir invoices

mkdir quotes

If you are going to create a directory, but its parent directory does not exist, you can use the -p (parents) option to have mkdir create all of the required parent directories too. In the following command, we are creating the "2019" directory inside the "yearly" directory inside the "quotes" directory. The "yearly" directory does not exist, but we can have mkdir create all the specified directories at once:

mkdir -p quotes/yearly/2019

22. **man:** This command shows the manual pages of another command.

man chown

23. **ls:** it lists the files and folders in the directory specified in the command line

To list the files and folders in the current directory:

ls

To list the files and folders in the current directory with a detailed listing use the -l (long) option:

ls -l

To use human-friendly file sizes include the -h (human) option:

ls -lh

To include hidden files use the -a (all files) option:

ls -lha

24. **finger:** This command gives short information about a user including time of the user's last login, the user's home directory and the user's accounts full name.

finger mary

25. **free:** this command gives a summery of the memory uses within your computer.

free -h

26. **grep:** It searces for lines which contains a search pattern.

To search a the word "train" in all text files in the current directory the command will be.

grep train *.txt

27. **groups:** This command prints which groups a user is a member of.

groups dave

28. **gzip:** The gzip command compresses files. By default, it removes the original file and leaves you with the compressed version. To retain both the original and the compressed version, use the -k (keep) option.

gzip -k core.c

29. **kill:** The kill command can terminate a process from the command line by providing the process ID (PID) of the process to kill.to find the PID of a process "ps" and "grep" command can be used.for example to get the PID of shutter process we can write:

ps -e | grep shutter.

Once we get the PID—1692 in this case—we can kill it as follows:

kill 1692

30. **find:** Use the find command to track down files that you know exist if you can't remember where you put them. You must tell find where to start searching from and what it is looking for. In this example, the . matches the current folder and the -name option tells find to look for files with a name that matches the search pattern.

find . -name *ones*

#WRITE A SHELL SCRIPT TO PRINT THE COMMAND LINE ARGUMENTS IN REVERSE ORDER

```bash
#!/bin/bash
if [ $# -eq 0 ]
then
echo "$0 num1,num2,numn"
exit 1
fi
x=""
echo -n "Numbers are:"
for n in $@
do echo -n $n
echo -n " "
x="$n $x"
done
echo ""
echo -n "Reverse order:"
echo $x
```

#WRITE A SHELL SCRIPT TO CHECK WHETHER THE GIVEN NUMBER IS PALINDROM OR NOT

```bash
#!/bin/bash
echo "Enter a number"
read n
r=0
rev=0
```

```bash
temp=$n
while(($n>0))
do
r=$(($n%10))
rev=$((((rev*10))+r))
n=$((n/10))
done
if(($temp==$rev))
then
echo "The enter number is palindrom"
else
echo "the enter number is not palindrom"
fi
```

#WRITE A SHELL SCRIPT TO SORT THE GIVEN ARRAY ELEMENTS IN ASCENDING ORDERS USING BUBBLE SORT

```bash
#!/bin/bash
clear
a=(10 8 20 100 12)
echo "Array in original order"
echo ${a[*]}

for((i=0; i<5;i++))
do
for((j=i;j<5-i-1;j++))
do
if ((${a[$j]}>${a[$((j+1))]}))
```

```
then
temp=${a[$j]}
a[$j]=${a[$((j+1))]}
a[$((j+1))]=$temp
fi
done
done
echo "Array in sorted order"
echo ${a[*]}
```

#WRITE A SHELL SCRIPT TO READ YOUR NAME , COLOUR AND PRINT THE SAME ON THE SCREEN

```
#!/bin/bash
clear
echo -n "what is your name?"
read name
clear
echo "Hello $name"
echo "what is your favorite color?"
read color
echo "$color is a good color"
echo "now saving that info"
echo "$name favorite color is $color" >> color.log
echo "date saved."
echo "Please press enter to continue"
read
clear
```

echo " Have a good day $name"

WRITE A SHELL SCRIPT TO PERFORM BINARY SEARCH ON A GIVEN ARRAY ELEMENTS

```
#!/bin/bash
clear
echo "read the limit"
read n
for((i=0;i<n;i++))
do
read m
a[i]=$m
done
for((i=1;i<n;i++))
do
for((j=0;j<n-i;j++))
do
if((${a[$j]}>${a[$((j+1))]}))
then
t=${a[$j]}
a[$j]=${a[$((j+1))]}
a[$((j+1))]=$t
fi
done
done
echo "sorted Array is"
echo ${a[*]}
```

```
echo "Enter the element to be searched"
read s
lb=0
ub=$((n-1))
c=0
while(($lb<$ub))
do
mid=$(((($lb+$ub))/2))
if(($s==${a[$mid]}))
then
c=1
break
elif(($s<${a[$mid]}))
then ub=$((mid-1))
else
lb=$((mid+1))
fi
done
if(($c==1))
then
echo "the Element is found at position $((mid+1))"
else
echo "the Element is not found in the list"
fi
```

#WRITE A SHELL SCRIPT TO ACCEPT ANY TWO FILE NAME AND CHECK THEIR FILE PERMISSION

```
#!/bin/bash
echo "Enter the file name"
read file
[ -w$file ]&&W="Write=yes"||W="Write=no"
[ -x$file ]&&X="Execute=yes"||X="Execute=no"
[ -r$file ]&&R="Read=yes"||R="Read=no"
echo "File Permission"
echo "$W"
echo "$X"
echo "$R"
```

WRITE A SHELL SCRIPT TO ILLUSTRATE THE CASE STATEMENT

```
#!/bin/sh
FRUIT="kiwi"
case "$FRUIT" in
   "apple") echo "Apple pie is quite tasty."
   ;;
   "banana") echo "I like banana nut bread."
   ;;
   "kiwi") echo "New Zealand is famous for kiwi."
   ;;
esac
```

#WRITE A SHELL SCRIPT TO DEMONSTRATE TERMINAL LOCKING

```sh
#!/bin/sh
echo "Enter a password"
read pass1
echo "Renter password"
read pass2
if [ $pass1 = $pass2 ]
then
    echo "Keyboard is locked"
    trap " " 20 30 15 9 2 1 3
    echo "Enter your password to unlock the screen"
    read upass
    while [ "$pass2" != "$upass" ]
    do
        echo "password is wrong !! "
        echo "Renter password"
        read upass
    done
else
    echo "Password doesn't match"
fi
```

SHELL SCRIPT TO ACCEPT THE VALID LOGIN NAME, IF THE LOGIN NAME IS VALID THEN PRINT ITS HOME DIRECTORY ELSE AN APPROPRIATE MESSAGE

```sh
#!/bin/sh
Clear
if [ $# -eq 0 ]
then
echo "No command line argument passed"
exit
fi
while [ $1 ]
do
cat /etc/passwd | cut -d ":" -f1 | grep "^$1" > temp
ck=`cat temp`
if [ "$ck" != "$1" ]
then
echo "ERROR:$1 is an invalid login name"
else
echo "Home Directory for $1 is"
echo `cat /etc/passwd | grep "^$1" | cut -d ":" -f6`
fi
shift
done
```

#WRITE A SHELL SCRIPT TO READ A FILE NAME AND CHANGE THE EXISTING FILE PERMISSIONS

```bash
#!/bin/bash
source /generic/utils/etc/environments/perm.conf
```

```
cd $ENVR
DIRS=`ls -l $ENVR | egrep '^d' | awk '{print $9}'`
for DIR in "${DIRS[@]}";
do
echo "$DIR"
echo "Which environment do you want?: "
echo -n "> "
read i
echo "Changing permissions now..."
sudochown -R $OWN:$GRP "$i" &&sudochmod -R $MOD1 "$i"
#cd $ENVR/$i
#sudochmod -R $MOD2 *
echo "Permissions are changed!"
done
```

#WRITE A SHELL SCRIPT THAT DISPLAY A MENU THAT CONSISTING OF OPTION TO DISPLAY DISK SPACE, THE CURRENT USER LOGGED IN , TOTAL MEMORY USAGE ETC.

```
#!/bin/bash
clear
echo "1. size of the hard disk"
echo "2. your user name"
echo "3.currently logged in users"
echo "4.current date and time"
echo "5. total memory usage"
echo "6.cpu load"
echo "7. exit"
```

```
echo
echo
echo "Enter your option"
read n
case $n in "1")
df –H | grep –vE '^Filesystem' | awk '{print $1 " " $2}';;
"2")
Echo "Your user name: $(echo users)";;
"3")
Echo "Currently login users:"
Who;;
"4")
Echo " current date and time:" $(date);;
"5")
Echo""
Free –m | awk 'NR==2{ printf"memory usage:%s%MB(%.2f%%)\n",
$3,$2,$3*100/$2}';;
"6")
Top –bn1 | grep load | awk '{printf"CPU Load:%.2f\n", $(NF-2)}';;
"7")
Echo "Have a nice day";;
*)
Echo "entered wrong option";;
esac
```

#WRITE A C-PROGRAM TO FORK A CHILD PROCESS AND EXECUTE THE GIVEN LINUX COMMANDS.

```c
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

int main() {

pid_tforkStatus;

forkStatus = fork();

/* Child... */

if      (forkStatus == 0) {

printf("Child is running, processing.\n");

sleep(5);

printf("Child is done, exiting.\n");

/* Parent... */

} else if (forkStatus != -1) {

printf("Parent is waiting...\n");

wait(NULL);

printf("Parent is exiting...\n");

} else {

perror("Error while calling the fork function");

}

return 0;

}
```

#WRITE A PROGRAM TO FORK A CHILD AND PRINT THE PROCESS ID OF PARENT AND CHILD PROCESS.

```c
int main()
{
int i;
printf("hello before fork \n");
printf("i : %d\n",i);
    i=fork();
printf("\n");
if(i==0)
    {
printf("Child has started\n\n");
printf("child printing first time \n");
printf("getpid : %d getppid : %d \n",getpid(),getppid());
sleep(5);
printf("\nchild printing second time \n");
printf("getpid : %d getppid : %d \n",getpid(),getppid());
    }
else
    {
printf("parent has started\n");
printf("getpid : %d  getppid : %d \n",getpid(),getppid());
printf("\n");
    }
printf("Hi after fork i : %d\n",i);
return 0;
}
```

#WRITE A C PROGRAM TO PROMPT USER FOR THE NAME OF THE ENVIORNMENT VARIABLE , CHECK IT'S VALIDITY AND PRINT THE MESSAGE

```c
#include <stdio.h>

int main(int argc, char *argv[], char * envp[])

{

    int i;

    for (i = 0; envp[i] != NULL; i++)

            printf("\n%s", envp[i]);

    getchar();

    return 0;

}
```

#WRITE A C PROGRAM TO RAED DETAILS OF N STUDENTS SUCH AS NAME, REGD NUMBER, AGE, SEMESTER. FIND THE ELDEST OF THEM AND PRINT HIS DETAILS

```c
#include <stdio.h>
struct student {
   char Name[50];
   int regd;
   char sem[8];
   int age;
} s[10];
```

```c
int main() {
    int i, Max,n,eld;
    printf("Enter the number of students(Maximum 10)\n");
    scanf("%d",&n);
    printf("Enter information of students:\n");
    // storing information
    for (i = 0; i < n; ++i) {
        printf("Enter name: ");
        scanf("%s", s[i].Name);
        printf("\nEnter regd no");
        scanf("%d", &s[i].regd);
        printf("Enter semester: ");
        scanf("%s", s[i].sem);
        printf("Enter age: ");
        scanf("%d", &s[i].age);
    }
    Max=s[0].age;
    for (i = 1; i < n; ++i) {
        if (s[i].age>Max)
        {
        Max=s[i].age;
        eld=i;
        }
    }
    printf("Displaying Information of eldest student:\n\n");
    // displaying information
printf("name: ");
```

```c
    puts(s[eld].Name);
 printf("\nRegd number: %d\n", s[eld].regd);
 printf("Semester: ");
  puts(s[eld].sem);
printf("Age: %d",s[eld].age);
 printf("\n");
 return 0;
}
```