



Laboratory Manual

On

OPERATING SYSTEM LAB

(For 4th Semester CSE/IT)

Prepared by:

Miss Barsha Subudhi Ray

Guest faculty (CSE/IT)

UCP Engineering School

Berhampur.

OPERATING SYSTEM LAB 4th SEMESTER

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

Steps to Install Git Bash

Follow the steps given below to install **Git Bash on Windows**:

Step 1: Download and Run the installer

The .exe file installer for Git Bash can be downloaded from
[“https://gitforwindows.org/”](https://gitforwindows.org/)

Once downloaded execute that installer, following window will occur.

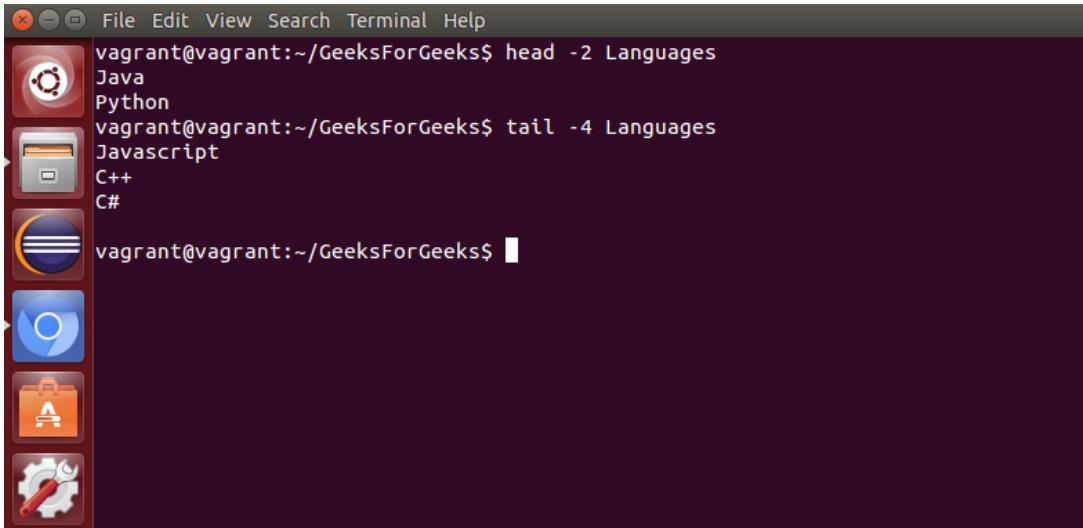
Basic Git Bash Commands

Here are some fundamental Git Bash commands to get you started:

1). Displaying the file contents on the terminal:

- **cat**: It is generally used to concatenate the files. It gives the output on the standard output.

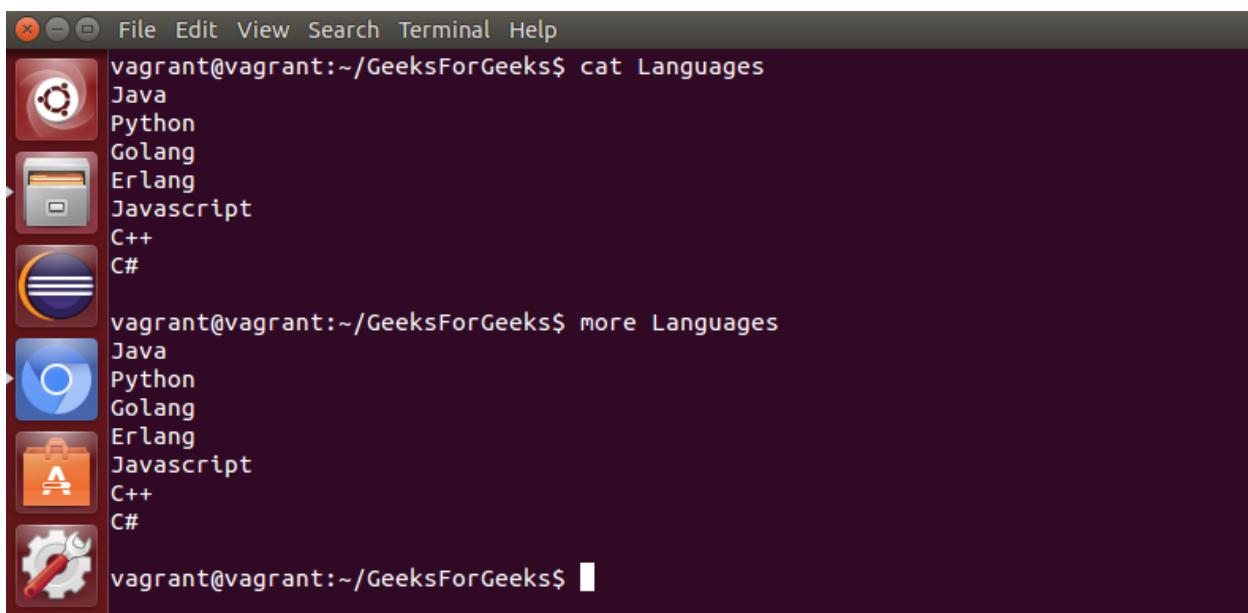
- **more**: It is a filter for paging through text one screenful at a time.
 - **less**: It is used to viewing the files instead of opening the file.Similar to *more* command but it allows backward as well as forward movement.
 - **head** : Used to print the first N lines of a file. It accepts N as input



```
vagrant@vagrant:~/GeeksForGeeks$ head -2 Languages
Java
Python
vagrant@vagrant:~/GeeksForGeeks$ tail -4 Languages
Javascript
C++
C#
vagrant@vagrant:~/GeeksForGeeks$
```

and the default value of N is 10.

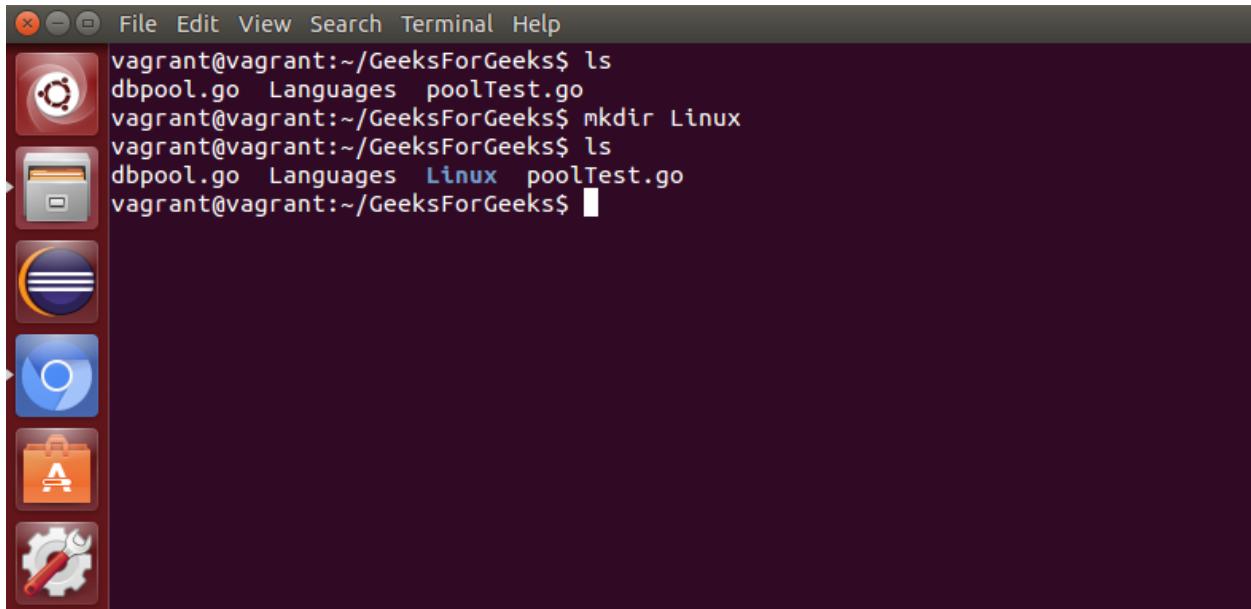
- **tail** : Used to print the last N-1 lines of a file. It accepts N as input and the default value of N is 10.



```
vagrant@vagrant:~/GeeksForGeeks$ cat Languages
Java
Python
Golang
Erlang
Javascript
C++
C#
vagrant@vagrant:~/GeeksForGeeks$ more Languages
Java
Python
Golang
Erlang
Javascript
C++
C#
vagrant@vagrant:~/GeeksForGeeks$
```

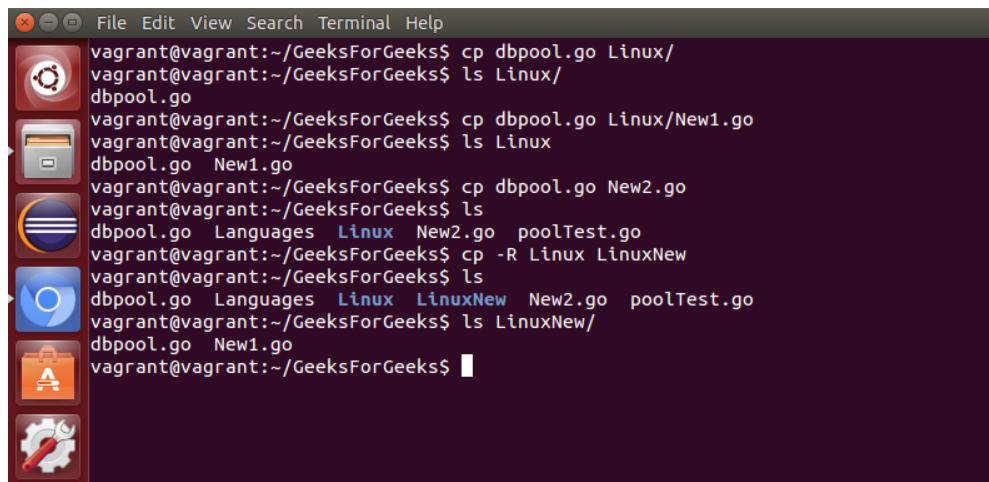
2). File and Directory Manipulation Commands:

- **mkdir** : Used to create a directory if not already exist. It accepts the directory name as an input parameter.



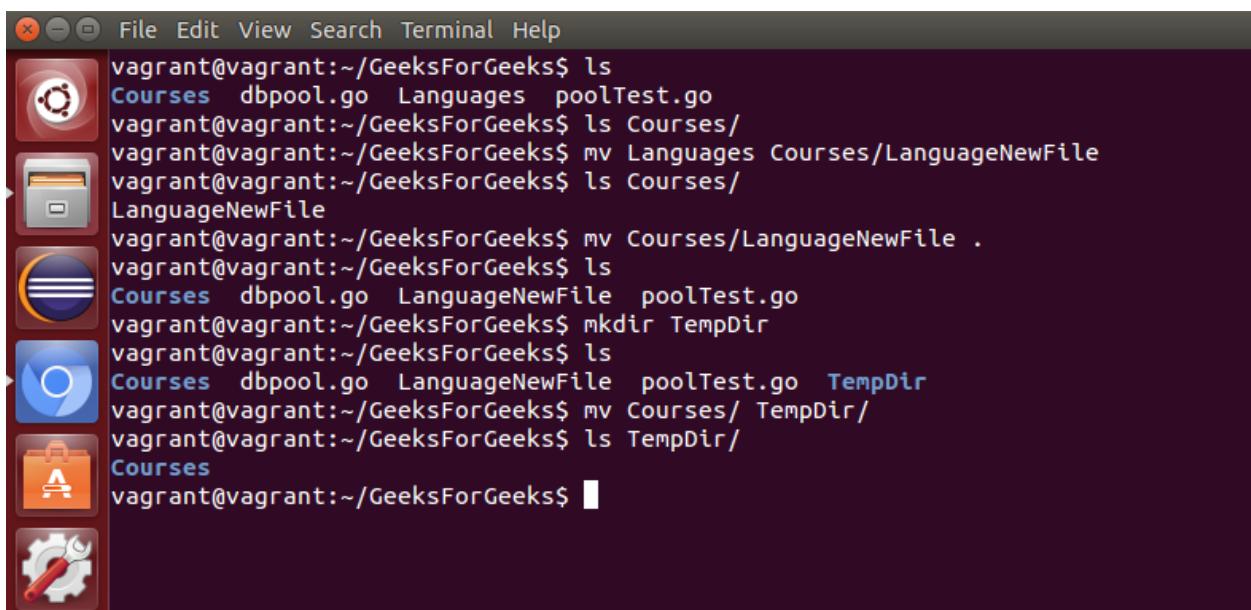
The screenshot shows a terminal window with a dark purple background. The window title bar includes icons for close, minimize, maximize, and a menu labeled "File Edit View Search Terminal Help". The terminal itself has a dark purple background and displays the following text:
vagrant@vagrant:~/GeeksForGeeks\$ ls
dbpool.go Languages poolTest.go
vagrant@vagrant:~/GeeksForGeeks\$ mkdir Linux
vagrant@vagrant:~/GeeksForGeeks\$ ls
dbpool.go Languages **Linux** poolTest.go
vagrant@vagrant:~/GeeksForGeeks\$ █
A vertical toolbar on the left side of the terminal window contains several icons: a red square with a white circle, a folder icon, a blue circular icon with horizontal lines, a blue square with a white circle, an orange folder icon with a white letter 'A', and a gear wrench icon.

- **cp** : This command will copy the files and directories from the source path to the destination path. It can copy a file/directory with the new name to the destination path. It accepts the source file/directory and destination file/directory.



```
vagrant@vagrant:~/GeeksForGeeks$ cp dbpool.go Linux/
vagrant@vagrant:~/GeeksForGeeks$ ls Linux/
dbpool.go
vagrant@vagrant:~/GeeksForGeeks$ cp dbpool.go Linux/New1.go
vagrant@vagrant:~/GeeksForGeeks$ ls Linux
dbpool.go New1.go
vagrant@vagrant:~/GeeksForGeeks$ cp dbpool.go New2.go
vagrant@vagrant:~/GeeksForGeeks$ ls
dbpool.go Languages Linux New2.go poolTest.go
vagrant@vagrant:~/GeeksForGeeks$ cp -R Linux LinuxNew
vagrant@vagrant:~/GeeksForGeeks$ ls
dbpool.go Languages Linux LinuxNew New2.go poolTest.go
vagrant@vagrant:~/GeeksForGeeks$ ls LinuxNew/
dbpool.go New1.go
vagrant@vagrant:~/GeeksForGeeks$
```

- **mv** : Used to move the files or directories. This command's working is almost similar to **cp** command but it deletes a copy of the file or directory from the source path.

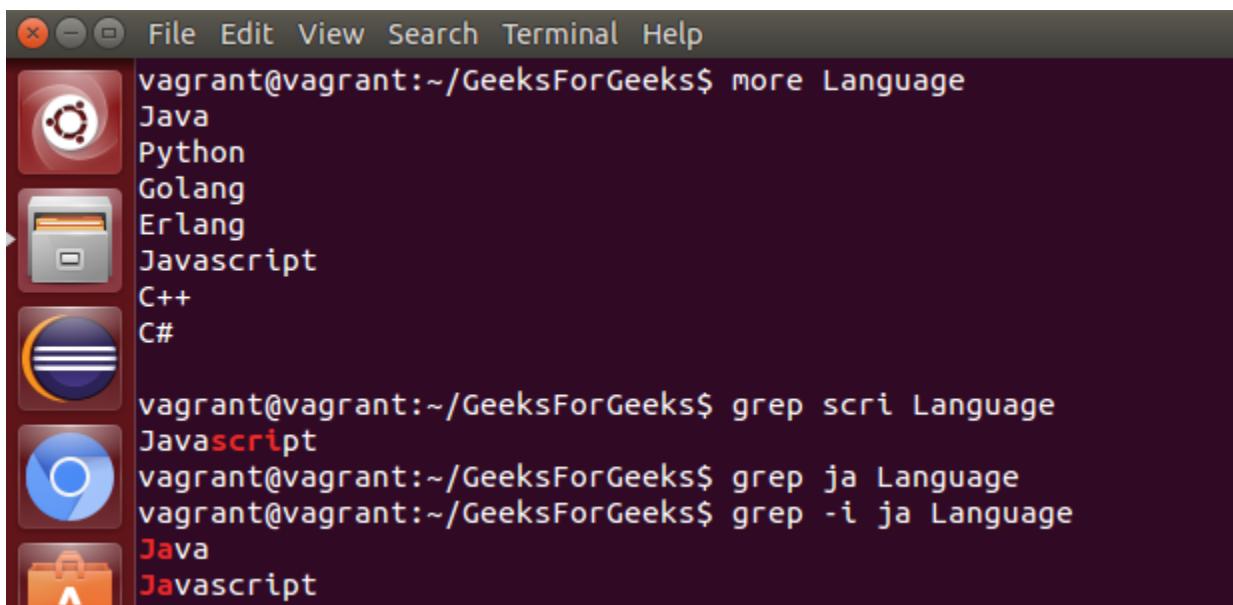


```
vagrant@vagrant:~/GeeksForGeeks$ ls
Courses dbpool.go Languages poolTest.go
vagrant@vagrant:~/GeeksForGeeks$ ls Courses/
vagrant@vagrant:~/GeeksForGeeks$ mv Languages Courses/LanguageNewFile
vagrant@vagrant:~/GeeksForGeeks$ ls Courses/
LanguageNewFile
vagrant@vagrant:~/GeeksForGeeks$ mv Courses/LanguageNewFile .
vagrant@vagrant:~/GeeksForGeeks$ ls
Courses dbpool.go LanguageNewFile poolTest.go
vagrant@vagrant:~/GeeksForGeeks$ mkdir TempDir
vagrant@vagrant:~/GeeksForGeeks$ ls
Courses dbpool.go LanguageNewFile poolTest.go TempDir
vagrant@vagrant:~/GeeksForGeeks$ mv Courses/ TempDir/
vagrant@vagrant:~/GeeksForGeeks$ ls TempDir/
Courses
vagrant@vagrant:~/GeeksForGeeks$
```

- **rm** : Used to remove files or directories.
- **touch** : Used to create or update a file.

3). Extract, sort, and filter data Commands:

- **grep** : This command is used to search for the specified text in a file.

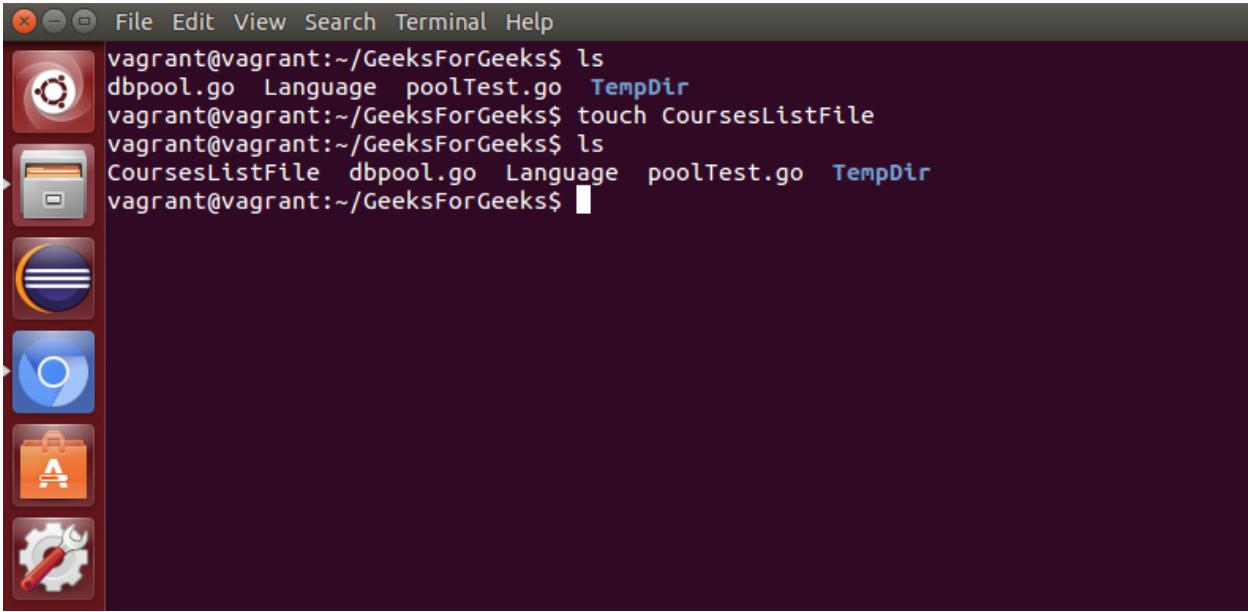


The screenshot shows a terminal window with a dark background and light-colored text. The window title bar includes 'File Edit View Search Terminal Help'. The terminal content displays the following commands and their outputs:

```
vagrant@vagrant:~/GeeksForGeeks$ more Language
Java
Python
Golang
Erlang
Javascript
C++
C#
vagrant@vagrant:~/GeeksForGeeks$ grep scri Language
Javascrit
vagrant@vagrant:~/GeeksForGeeks$ grep ja Language
vagrant@vagrant:~/GeeksForGeeks$ grep -i ja Language
Java
Javascript
```

Icons for various programming languages are visible on the left side of the terminal window.

```
vagrant@vagrant:~/GeeksForGeeks$ ls  
dbpool.go Language poolTest.go TempDir  
vagrant@vagrant:~/GeeksForGeeks$ touch CoursesListFile  
vagrant@vagrant:~/GeeksForGeeks$ ls  
CoursesListFile dbpool.go Language poolTest.go TempDir  
vagrant@vagrant:~/GeeksForGeeks$
```

A screenshot of a Linux desktop environment, specifically Ubuntu, showing a terminal window. The terminal window has a dark purple background and contains the following text:

```
vagrant@vagrant:~/GeeksForGeeks$ ls  
dbpool.go Language poolTest.go TempDir  
vagrant@vagrant:~/GeeksForGeeks$ touch CoursesListFile  
vagrant@vagrant:~/GeeksForGeeks$ ls  
CoursesListFile dbpool.go Language poolTest.go TempDir  
vagrant@vagrant:~/GeeksForGeeks$
```

The terminal window has a standard window title bar with icons for close, minimize, and maximize. To the left of the terminal window, there is a vertical dock containing several icons: a red square icon, a folder icon, a blue circular icon, a blue square icon with a white letter 'A', and a gear wrench icon.

- **sort** : This command is used to sort the contents of files.

```
vagrant@vagrant:~/GeeksForGeeks$ more Language
Java
Python
Golang
Erlang
Javascript
C++
C#
vagrant@vagrant:~/GeeksForGeeks$ sort Language
C#
C++
Erlang
Golang
Java
Javascript
Python
vagrant@vagrant:~/GeeksForGeeks$ sort -r Language
Python
Javascript
Java
Golang
Erlang
C++
C#
vagrant@vagrant:~/GeeksForGeeks$ sort -R Language
C++
Golang
Java
Javascript
Erlang
Python
C#
```

- **wc** : Used to count the number of characters, words in a file.

```
vagrant@vagrant:~/GeeksForGeeks$ more CoursesListFile
Advanced Java Course
Python
Spring Framework
hybernate
vagrant@vagrant:~/GeeksForGeeks$ wc CoursesListFile
4 7 55 CoursesListFile
vagrant@vagrant:~/GeeksForGeeks$ wc -l CoursesListFile
4 CoursesListFile
vagrant@vagrant:~/GeeksForGeeks$ wc -w CoursesListFile
7 CoursesListFile
vagrant@vagrant:~/GeeksForGeeks$ wc -m CoursesListFile
55 CoursesListFile
vagrant@vagrant:~/GeeksForGeeks$
```

```
C++
C#
vagrant@vagrant:~/GeeksForGeeks$ cut -c 2-4 Language
ava
yth
ola
rla
ava
++
#
vagrant@vagrant:~/GeeksForGeeks$
```

- **cut** : Used to cut a specified part of a file.

4). Basic Terminal Navigation Commands:

- **ls** : To get the list of all the files or folders.
- **ls -l**: Optional flags are added to **ls** to modify default behavior, listing contents in extended form -**l** is used for “**long**” **output**
- **ls -a**: Lists of all files including the hidden files, add **-a flag**
- **cd**: Used to change the directory.
- **du**: Show disk usage.
- **pwd**: Show the present working directory.
- **man**: Used to show the manual of any command present in Linux.
- **rmdir**: It is used to delete a directory if it is empty.
- **ln file1 file2**: Creates a physical link.
- **ln -s file1 file2**: Creates a symbolic link.
- **locate**: It is used to locate a file in Linux System
- **echo**: This command helps us move some data, usually text into a file.
- **df**: It is used to see the available disk space in each of the partitions in your system.
- **tar**: Used to work with tarballs (or files compressed in a tarball archive)

5). File Permissions Commands: The **chmod** and **chown** commands are used to control access to files in UNIX and Linux systems.

- **chown** : Used to change the owner of the file.
- **chgrp** : Used to change the group owner of the file.
- **chmod** : Used to modify the access/permission of a user.

OPERATING SYSTEM LAB 4th SEMESTER

Exp1. Write a Shell script to print the command line arguments in reverse order.

Ans.

```
#!/bin/bash
if [ $# -eq 0 ]
then
    echo "no argumnets given"
    exit
fi
echo "arguments no. $#"
echo "arguments value $*"
echo "in reverse"
rev=""
for i in $*
do
    rev=$i" "$rev
done
echo $rev
```

```
UCPES11@UCPES MINGW64 ~/Desktop/akshya
$ chmod +x reverse.sh
UCPES11@UCPES MINGW64 ~/Desktop/akshya
$ bash reverse.sh
no argumnets given
```

Exp2. Write a Shell script to check whether the given number is palindrome or not.

Ans.

```
echo "Enter a number: "
read number

reverse=0
original=$number

while [ $number -ne 0 ]
do
    remainder=$(( $number % 10 ))
    reverse=$(( $reverse * 10 + $remainder ))
    number=$(( $number / 10 ))
done

if [ $original -eq $reverse ]
then
    echo "$original is a palindrome."
else
    echo "$original is not a palindrome."
fi
```

```
UCPES11@UCPES MINGW64 ~/Desktop/akshya
$ chmod +x palindrome.sh

UCPES11@UCPES MINGW64 ~/Desktop/akshya
$ bash palindrome.sh
Enter a number:
123543
123543 is not a palindrome.
```

Exp3. Write a Shell script to sort the given array elements in ascending order using bubble sort.

Ans.

```
arr=(11 44 55 22 33)
echo "Array in original order"
echo ${arr[*]}
for((i=0; i<5; i++))
do
for((j=0; j<5-i-1;j++))
do
if [ ${arr[j]} -gt ${arr[$((j+1))]} ]
then
temp=${arr[j]}
arr[$j]=${arr[$((j+1))]}
arr[$((j+1))]=$temp
fi
done
done
echo "Array in sorted order :"
echo ${arr[*]}
```

```
UCPES11@UCPES MINGW64 ~/Desktop/akshya
$ bash bubblesort.sh
Array in original order
11 44 55 22 33
Array in sorted order :
11 22 33 44 55
```

Exp4 . Write a Shell script to perform sequential search on a given array elements.

Ans

echo Enter the number of elements:

read n

echo Enter the array elements:

for ((i=1; i<=n; i++))

do

 read a[\$i]

done

echo Enter the element to be searched:

read item

j=1

while [\$j -lt \$n -a \$item -ne \${a[\$j]}]

do

 j=`expr \$j + 1`

done

if [\$item -eq \${a[\$j]}]

then

 echo \$item is present at location \$j

else

 echo "\$item is not present in the array."

Fi

```
UCPES11@UCPES MINGW64 ~/Desktop/akshya
$ bash sequential.sh
Enter the number of elements:
6
Enter the array elements:
34
54
3
35
22
98
Enter the element to be searched:
22
22 is present at location 5
```

Exp5. Write a Shell script to perform binary search on a given array elements.

Ans

```
echo "Enter the limit:"  
read n  
echo "Enter the numbers"  
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]} -gt ${a[$j+1]} ]  
then  
t=${a[$j]}  
a[$j]=${a[$j+1]}  
a[$j+1]=$t  
fi  
done  
done  
echo "Sorted array is"  
for(( i=0; i<n; i++ ))  
do
```

```
echo "${a[$i]}"

done

echo "Enter the element to be searched :"

read s

l=0

c=0

u=$((n-1))

while [ $l -le $u ]

do

mid=$((( $l+$u ))/2))

if [ $s -eq ${a[$mid]} ]

then

c=1

break

elif [ $s -lt ${a[$mid]} ]

then

u=$((mid-1))

else

l=$((mid+1))

fi

done

if [ $c -eq 1 ]

then

echo "Element found at position $((mid+1))"

else
```

```
echo "Element not found"
```

```
fi
```

```
UCPES11@UCPES MINGW64 ~/Desktop/akshya
$ bash binarysearch.sh
"Enter the limit:"
4
"Enter the numbers"
3
6554
33
56
"Sorted array is"
"3"
"33"
"56"
"6554"
"Enter the element to be searched :"
6554
"Element found at position 4"
```

Exp 6. Write a Shell script to accept any two file names and check their file permissions.

Ans.

```
echo "Enter 1st file names"
```

```
read file1
```

```
echo "Enter 2nd file names"
```

```
read file2
```

```
if [ -f $file1 -a -f $file2 ]
```

```
then
```

```
    echo "$file1 and $file2 exists "
```

```
else
```

```
    echo "Files doesn't existZ"
```

```
exit
```

```
fi
```

```
if [ -x $file1 ]
then
    echo "$file1 has execute permission."
fi

if [ -r $file1 ]
then
    echo "$file1 has read permission."
fi

if [ -w $file1 ]
then
    echo "$file1 has write permission."
fi

echo "-----"
if [ -x $file2 ]
then
    echo "$file2 has execute permission."
fi

if [ -r $file2 ]
then
    echo "$file2 has read permission."
fi

if [ -w $file2 ]
then
    echo "$file2 has write permission."
fi
```

```
kirrtt@DESKTOP-HUU017E MINGW64 ~
$ cd desktop

kirrtt@DESKTOP-HUU017E MINGW64 ~/desktop
$ chmod +x Ex6.sh

kirrtt@DESKTOP-HUU017E MINGW64 ~/desktop
$ ./Ex6.sh file1.txt file2.txt
The permissions for file1.txt are: -rw-r--r--
./Ex6.sh: line 25: check_permissions file2.txt: command not found

kirrtt@DESKTOP-HUU017E MINGW64 ~/desktop
$ |
```

Exp7. 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'.

Ans

```
#!/bin/bash
```

Read-p "Enter the path name :"path

IFS = 'I' read – ra dirs. <<< "\$path"

For dirin "\${dirc[@]}"

Do

If [d"\$dir"]

Then

Echo "directory \$dir already exists"

Else

#create the directory

Mkdir "\$dir"

echo "Directory '\$dir' created"

Fi

#change to the new directory

```
Cd "$dir"
```

```
done
```



```
HP@DESKTOP-6LEL03D MINGW64 ~/Desktop
$ bash os7.sh
Enter the path name: C:\Users\HP\Desktop
Path created successfully!
```

Exp8. Write a Shell script to illustrate the case-statement.

Ans

```
#!/bin/bash

#prompt the user for input
echo "Enter a Fruit"

read fruit

# use the case statement to perform different actions based on the
input case $ Fruit in

"apple")
    echo" you selected apple"
    echo "Apples are red"
;;
"banana")
    echo "you setected banana"
```

```
echo "oranges are orange"  
;;  
*)  
echo "unknown fruit"  
;;  
esac
```

```
HP@DESKTOP-6LEL03D MINGW64 ~/Desktop  
$ bash os8.sh  
Select an option:  
1. Say Hello  
2. Say Goodbye  
3. Exit  
1  
Hello!
```

Exp9. Write a Shell script to accept the file name as arguments and create another shell script, which recreates these files with its original contents.

Ans

```
#!/bin/bash

if [ $# -eq 0]; then

echo "Please provide a file name as an argument"

read file name

fi

#create a new shell script with the provided file touch $ l.sh

# Add she bang to the new script

echo "#!/bin/bash">>> $ l.sh

#mark the new script executable

Chmode +x $ l.sh

echo" New shell script created with name & File.sh"
```

```
HP@DESKTOP-6LEL03D MINGW64 ~/Desktop
$ bash restore_files.sh file1.txt file2.txt
File 'file1.txt' not found. Skipping...
File 'file2.txt' not found. Skipping...
```

Exp10. Write a Shell script to demonstrate Terminal locking.

Ans

```
#!/bin/bash

# Define the lock file

lock-file = " 1 + mp/my_Script lock"

if [-e "$ lock-file "]; +1

then

echo "Another instances of the script is already running. Existing"

exit 1

Fi

touch "$ lock-file"

echo "performing Same work...."

Sleep 5

rom "$ lock-file"

echo "Script Completed lock released"
```

```
HP@DESKTOP-6LEL03D MINGW64 ~/Desktop
$ bash os10.sh
Creating lock file...
Performing some task...
Removing lock file...
Script execution completed.
```

Exp 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.

Ans

```
#!/bin/bash

# prompt the user two enter a login name read-p" Enter a valid login.

name:" login-name

# Check if the user exists

if id-U" $ login-name "> /dev/null 2> $1; then

# if the user exists print the name directory

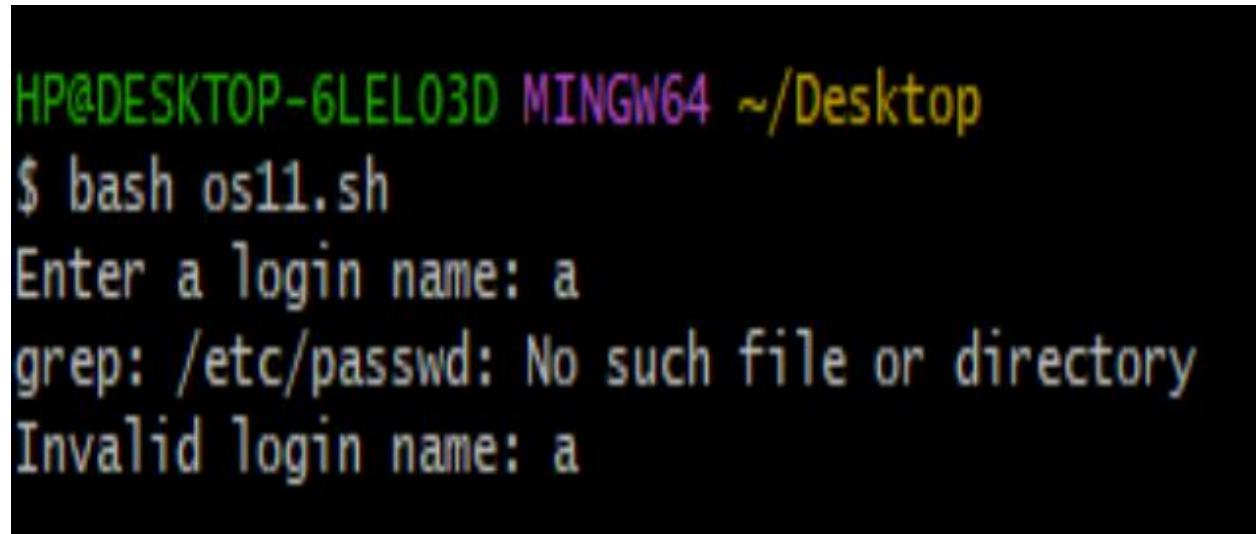
echo "Home directory Form user & login name is :" $(getent password $ Login name login d"-fb")"

else

# if the user does not exit display an error

echo" user $ login-name does not exist."

fi
```



The terminal window shows the command \$ bash os11.sh being run. It prompts for a login name 'a' and then displays an error message 'grep: /etc/passwd: No such file or directory' followed by 'Invalid login name: a'.

```
HP@DESKTOP-6LEL03D MINGW64 ~/Desktop
$ bash os11.sh
Enter a login name: a
grep: /etc/passwd: No such file or directory
Invalid login name: a
```

Exp12. Write a Shell script to read a file name and change the existing file permissions.

Ans

```
#!/bin/bash

Read.p "Enter file name: " file name

#Check if file exists

if [ $ -I "$ file name" ]; then

echo "file not round".

exit 1

fi

#prompt user to enter now file permissions

read-p"(Enter newfile permissions lin octalformat): "permission

# change file permission

Chmod "$ permission" " $ file "name"

echo "file permission changed to $ permission for & file & file name"
```

```
HP@DESKTOP-6LEL03D MINGW64 ~/Desktop
$ bash os12.sh
Enter file name: os12.sh
Enter new file permissions (in octal format, e.g., 644)
chmod: invalid mode: '445.sh'
Try 'chmod --help' for more information.
Permissions for file 'os12.sh' changed to '445.sh'.
```

Exp13 Write a Shell script to print current month calendar and to replace the current day number by '*' or '**' respectively.

Ans

```
#!/bin/bash

read-p" Enter file name:" file name

#Check if file exists

if [ -f "$ file name" ]; then
```

```

echo "File not found"

exit 1

fi

#prompt user to enter now file permissions

read-p" Enter newfile permissions (in octalformal): "permission

#Change File permission

Chmod "$ permission" " & File name"

echo "file permission changed to $ permissions fore $ file $ file name"

```

```

HP@DESKTOP-6LELO3D MINGW64 ~/Desktop
$ bash os13.sh

```

April 2024
Su Mo Tu We Th Fr Sa
1 2 3 4 5 6
7 8 9 10 11 ** 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30

Exp14. 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.)

Ans

```

#!/bin/bash

display-disk-space(){

df.h

}

display-users(){


```

```
who
}

display-memory-usuage){

Free-M

}

# display menu and prompt fore user input while input

While true; do

echo "Menu:"

echo "1. Display the disk space"

echo "2. Display current users logged in"

echo " 3. Display total memory ususage"

echo "4.Exit"

Read-p"Enter your choice; "Choice

Case $ Choice in

(1)

display-disk-space

;;

(2)

display-users

;;

(3)

display-memory-usuage ::

(4)

exit

;;
```

```

*)
echo "invalid choice"
;;
esac
echo "press Enter to continue"
read
done

```

```

HP@DESKTOP-6LELO3D MINGW64 ~/Desktop
$ bash os14.sh
Select the Option From below:

[ 1 ] For Only the Disk-Name and Used-Space
[ 2 ] For Only the Disk-Name and its Size
[ 3 ] To print Disk-Name and File-System
[ 4 ] To see all fields in DiskUsage

2
Filesystem          Size
C:/Program Files/Git 472G
F:                  4.9G

```

Exp15. Write a C-program to fork a child process and execute the given Linux commands.

Ans.

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(){

pid_t pid;

```

```

Fork a child process

pid: fork():

if (pid <6){ // Error

F print f (stderr "Fork Failed \h");

exit(1); }

else if (pid == 0) { // Child process

System ("/s-1");

System (" echo System Hello World");}

Else { // parent process

Wait (NULL);

print f ("child process complete \h");

}

return 0;

}

```

```

HP@DESKTOP-6LEL03D MINGW64 ~/Desktop
$ bash os15.sh

total 20
-rw-r--r-- 1 user user 285 Apr 16 09:30 fork_exec.c
-rwxr-xr-x 1 user user 8608 Apr 16 09:30 fork_exec

```

Exp16. Write a C-program to fork a child process, print owner process ID and its parent process ID

Ans

```

#include <stdio.h>

#include <unistd.h>

int main()

pid = T Pid;

```

```

Pid = fork ();

if (pid <0){

    l print f (Sindr "fork Failed");

    return 1;

}

else if (pid==0){

printf("Child process: owner pin=%d,Parrent pin = %d\h". get pin().

get ppid());{

else{

printf ("parent process: owner p/n = % a, parent pin = %d\h",pide, get pid();

gey ppid());

}

Return 0;

}

```

```

HP@DESKTOP-6LEL03D MINGW64 ~/Desktop
$ bash os16.sh

Parent process PID: 1234
Parent process PPID: 5678
Child process PID: 1235
Child process PPID: 1234

```

Exp17. Write a C-program to prompt the user for the name of the environment variable, check its validity and print an appropriate message.

Ans

```

#include<stdio.h>

#include <stdlib.h>

#include <string.h>

int main(){


```

```

Chae env_var [100];

Char* value

printf ("Please enter the name of environment variable");

Scanf ("%s", evn-var);

value = get ernv(env-var);

if (value == NULL){

printf("The value of the environment Variable \"%s\" 's'%s\n");

env_var,value;

}

return o;
}

```

```

HP@DESKTOP-6LEL03D MINGW64 ~/Desktop
$ bash os17.sh

Enter the name of the environment variable: PATH
The value of PATH is: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin

```

Exp18. 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.

Ans

```

#include <studio.h>

#include <string.h>

#define MAX_STUDENT 100

Struct student{

Chare name [50];

int reg_no;

int semester;

```

```
int age;
};

int main(){
int n,i;
Struct Student students [MAX-STUDENTS];
Struct student eldest_Student;
printf("Enter the number of Students: ");
Scanf ("%d");
for(i=0,i<h; i++){
printf ("In Enter details of student %d: \h", i+1);
print("Name");
Scanf ("%s", Students [i].name);
printf("Registration Number: ");
Scanf("% a", & student [ ]. Reg_no);
printf ("semester: ");
scanf("%d", & students [i]. Semester);
printf ("Age:");
Scanf("%d", & student [1].age);
}
eldest _Student=students [0];
for (i=1;i<h; i++) {
is (Student [i].age> eldest_Student .age){
eldest-student = students [i];
}
printf ("In Details of the eldest Student: /n");
}
```

```
printf ("Name: %s\n", eldest-student Name);
printf ("Registration Number: %d \n" eldest student.reg.no);
printf("semester: %d\n", eldest-Student-semester);
printf ("Age: %d\n", eldest student.ages;
return 0;
}
```

```
HP@DESKTOP-6LEL03D MINGW64 ~/Desktop
$ bash os18.sh

Enter the number of students: 3
Enter details for student 1:
Name: John
Registration Number: 1001
Semester: 3
Age: 21
Enter details for student 2:
Name: Emma
Registration Number: 1002
Semester: 4
Age: 22
```