SKILL ORIENTED COURSE-II

CSSL2-LINUX PROGRAMMING

S. No.	Name of Experiment	Page No.	Date of Experiment
1	Module 1 - Directory Related Utilities		
2	Module 2 - File Handling and Text Processing		
3	Module 3 - Disk Utilities, Backup and Other Utilities		
4	Module 4 – Programmable Text Processing		
5	Module 5 – Shell Scripting		
6	Module 6 – File Management System Calls		
7	Module 7 – Process Management System Calls		
8	Module 8 – Operations on Signals		
9	Module 9 – Inter Process Communication		

Module-1

Directory Related Utilities

1.***pwd(Present Working Directory)***: It is used to print the full path of the present working directory starting from the root.

```
y22cs139@rvrcse:~$ pwd
/users/y22cs/y22cs139
y22cs139@rvrcse:~$
```

2.***mkdir(Make Directory)***: It is used to create directory or directories if not exists.

Before:

```
y22cs139@rvrcse:~$ ls
ds.c  examples.desktop  post.c  postfix.c
y22cs139@rvrcse:~$
```

After:

```
y22cs139@rvrcse:~$ mkdir OMPR
y22cs139@rvrcse:~$ ls
OMPR  ds.c  examples.desktop  post.c  postfix.c
y22cs139@rvrcse:~$
```

3.***ls(List)***: ls - list directory contents.

a) ls: This Command lists only the viewable contents of the directory.

```
y22cs139@rvrcse:~$ ls
OMPR  ds.c  examples.desktop  post.c  postfix.c
y22cs139@rvrcse:~$
```

b) ls –l: This Command is used to print a larger list with all the permissions, users and owner names.

```
y22cs139@rvrcse:~$ ls -l
total 28
drwxrwxr-x 2 y22cs139 y22cs139 4096 Jan 25 11:41 OMPR
-rw-rw-r-- 1 y22cs139 y22cs139 2409 Sep 15 12:59 ds.c
-rw-r--r-- 1 y22cs139 y22cs139 8980 Apr 16  2018 examples.desktop
-rw-rw-r-- 1 y22cs139 y22cs139  499 Nov  3 13:18 post.c
-rw-rw-r-- 1 y22cs139 y22cs139  689 Nov  3 14:13 postfix.c
y22cs139@rvrcse:~$
```

c) ls –a: This Command is used to print the hidden files that start with "." in the directory.

```
y22cs139@rvrcse:~$ ls -a
.  ..  .bash_history  .bash_logout  .bashrc  .cache  .gnupg  .profile  .viminfo  OMPR  ds.c  examples.desktop  post.c  postfix.c
y22cs139@rvrcse:~$
```

4.***cd(Change Directory)***: It is used to change the directory to the home directory by default or to a specified directory.

Before:

```
y22cs139@rvrcse:~$ pwd
/users/y22cs/y22cs139
y22cs139@rvrcse:~$
```

After:

```
y22cs139@rvrcse:~$ cd OMPR
y22cs139@rvrcse:~/OMPR$ pwd
/users/y22cs/y22cs139/OMPR
y22cs139@rvrcse:~/OMPR$
```

5.***rmdir(Remove Directory)***: It is used to remove directory only if they are empty.

Before:

```
y22cs139@rvrcse:~$ ls
OMPR  ds.c  examples.desktop  post.c  postfix.c
y22cs139@rvrcse:~$
```

After:

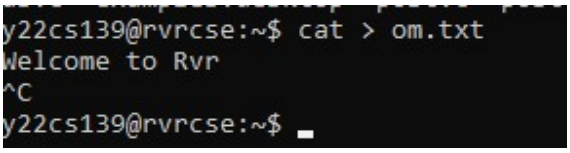
```
y22cs139@rvrcse:~$ rmdir OMPR
y22cs139@rvrcse:~$ ls
ds.c  examples.desktop  post.c  postfix.c
y22cs139@rvrcse:~$
```

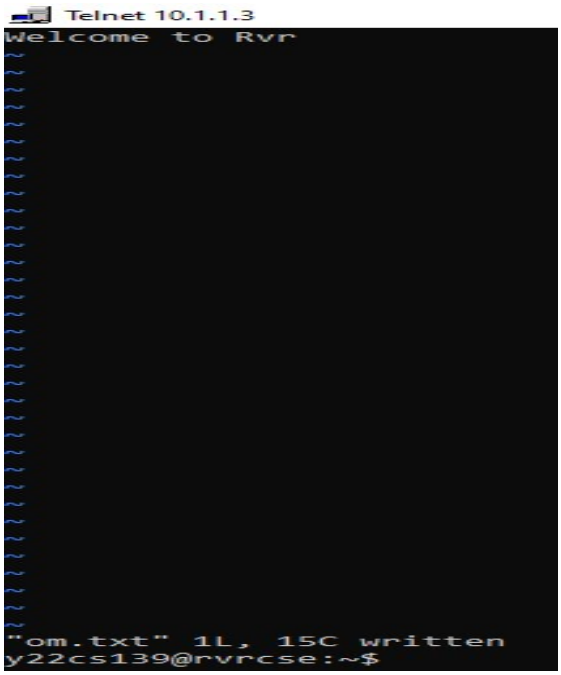
Module-2

File Handling And Processing Utilities.

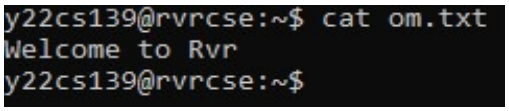
1.***cat(Concatenate)***: It is used to concatenate the files and print all the standard output.

a)cat> filename: It creates a new file and writes contents to it.





b)cat filename: It prints the contents of the file on Standard Output.



c)cat >> filename: It is used to append external content to the file.

Before:

```
y22cs139@rvrcse:~$ cat om.txt
Welcome to Rvr
y22cs139@rvrcse:~$
```

After:

```
y22cs139@rvrcse:~$ cat >> om.txt
Guntur,Andhra Pradesh.
^C
y22cs139@rvrcse:~$ cat om.txt
Welcome to Rvr
Guntur,Andhra Pradesh.
y22cs139@rvrcse:~$
```

2.***cp(Copy)***: It is used to copy files or directories from a source to destination in the file system.

```
y22cs139@rvrcse:~$ cp om.txt ompr.txt
y22cs139@rvrcse:~$ cat ompr.txt
Welcome to Rvr
Guntur,Andhra Pradesh.
y22cs139@rvrcse:~$
```

3.***mv(Move)***: It is used to move a file from one location to another or to rename a file.

Before:

```
y22cs139@rvrcse:~$ ls
ds.c  examples.desktop  om.txt  ompr.txt  post.c  postfix.c
y22cs139@rvrcse:~$
```

After:

```
y22cs139@rvrcse:~$ mv om.txt oruganti.txt
y22cs139@rvrcse:~$ ls
ds.c  examples.desktop  ompr.txt  oruganti.txt  post.c  postfix.c
y22cs139@rvrcse:~$
```

4.***rm(Remove):*** It is used to remove the specified file.

Before:

```
y22cs139@rvrcse:~$ ls
ds.c  examples.desktop  ompr.txt  oruganti.txt  post.c  postfix.c
y22cs139@rvrcse:~$
```

After:

```
y22cs139@rvrcse:~$ rm oruganti.txt
y22cs139@rvrcse:~$ ls
ds.c  examples.desktop  ompr.txt  post.c  postfix.c
y22cs139@rvrcse:~$
```

5.***ln –s (Link):*** It is used to create symbolic links between two files.

Before:

```
y22cs139@rvrcse:~$ cat oruganti.txt
Hi Everyone.
y22cs139@rvrcse:~$ ls
ds.c  examples.desktop  ompr.txt  oruganti.txt  post.c  postfix.c
y22cs139@rvrcse:~$
```

After:

```
y22cs139@rvrcse:~$ ln -s oruganti.txt paparao.txt
y22cs139@rvrcse:~$ ls
ds.c  examples.desktop  ompr.txt  oruganti.txt  paparao.txt  post.c  postfix.c
y22cs139@rvrcse:~$ cat paparao.txt
Hi Everyone.
y22cs139@rvrcse:~$
```

6.***sort:*** It is used to print the content in the existed txt file in a sorted order in the standard output.This is used to print sorted content only for txt files that contains numbers or characters.

.txt file containing numbers:

```
y22cs139@rvrcse:~$ cat sort1.txt
999
656
932
65
1
y22cs139@rvrcse:~$ sort sort1.txt
1
65
656
932
999
y22cs139@rvrcse:~$
```

.txt file containing characters:

```
y22cs139@rvrcse:~$ cat sort2.txt
q
r
t
w
a
k
y22cs139@rvrcse:~$ sort sort2.txt
a
k
q
r
t
w
y22cs139@rvrcse:~$
```

7.***unlink***: It is used to remove the symbolic links between files.

For Linking:

```
y22cs139@rvrcse:~$ ln -s 1.txt 2.txt
y22cs139@rvrcse:~$ ls -l
total 44
-rw-rw-r-- 1 y22cs139 y22cs139   13 Feb  1 12:01 1.txt
lrwxrwxrwx 1 y22cs139 y22cs139    5 Feb  1 12:07 2.txt -> 1.txt
-rw-rw-r-- 1 y22cs139 y22cs139 2409 Sep 15 12:59 ds.c
-rw-r--r-- 1 y22cs139 y22cs139 8980 Apr 16  2018 examples.desktop
-rw-rw-r-- 1 y22cs139 y22cs139   38 Jan 25 12:47 ompr.txt
-rw-rw-r-- 1 y22cs139 y22cs139   13 Jan 25 12:47 oruganti.txt
-rw-rw-r-- 1 y22cs139 y22cs139  499 Nov  3 13:18 post.c
-rw-rw-r-- 1 y22cs139 y22cs139  689 Nov  3 14:13 postfix.c
-rw-rw-r-- 1 y22cs139 y22cs139   17 Feb  1 11:25 sort1.txt
-rw-rw-r-- 1 y22cs139 y22cs139   12 Feb  1 11:49 sort2.txt
```

For Unlinking:

```
y22cs139@rvrcse:~$ unlink 2.txt
y22cs139@rvrcse:~$ ls -l
total 44
-rw-rw-r-- 1 y22cs139 y22cs139   13 Feb  1 12:01 1.txt
-rw-rw-r-- 1 y22cs139 y22cs139 2409 Sep 15 12:59 ds.c
-rw-r--r-- 1 y22cs139 y22cs139 8980 Apr 16  2018 examples.desktop
-rw-rw-r-- 1 y22cs139 y22cs139   38 Jan 25 12:47 ompr.txt
-rw-rw-r-- 1 y22cs139 y22cs139   13 Jan 25 12:47 oruganti.txt
-rw-rw-r-- 1 y22cs139 y22cs139  499 Nov  3 13:18 post.c
-rw-rw-r-- 1 y22cs139 y22cs139  689 Nov  3 14:13 postfix.c
-rw-rw-r-- 1 y22cs139 y22cs139   17 Feb  1 11:25 sort1.txt
-rw-rw-r-- 1 y22cs139 y22cs139   12 Feb  1 11:49 sort2.txt
y22cs139@rvrcse:~$
```

8.***head***: It is used to print the specified number of lines of content of a file from starting.

```
y22cs139@rvrcse:~$ cat head1.txt
aaa
bbb
iii
ooo
ppp
yyy
jjj
ooo
eee
y22cs139@rvrcse:~$ head -3 head1.txt
aaa
bbb
iii
```

9.***tail***: It is used to print the specified number of lines of the content of a file from the bottom.

```
y22cs139@rvrcse:~$ cat tail1.txt
aaa
bbb
ccc
ddd
hhh
ggg
y22cs139@rvrcse:~$ tail -3 tail1.txt
ddd
hhh
ggg
y22cs139@rvrcse:~$
```

10.***find***: It is used to find files or directories in a specified directory and a specified name expression.

```
y22cs139@rvrcse:~$ find -name orug*
./oruganti.txt
y22cs139@rvrcse:~$
```

If you have similar name files more than one use “find filenamecode” as command.

11.***more***: It is used for paging through text one-screen full at a time.

```
y22cs139@rvrcse:~$ cat more1.txt
hi welcome
good morning
welcome to RVR
this is telnet
this this this
that that that
aaa
b
c
d
```

```
o
p
q
r
s
t
u
v
--More--(49%)[Press space to continue, 'q' to quit.]
```

```
R
S
T
U
V
W
X
--More--(71%)[Press space to continue, 'q' to quit.]
```

```
18
19
20
21
22
23
24
25
26
--More--(98%)[Press space to continue, 'q' to quit.]
```

12.***w***: It is used to display the information about the user currently on the machine & their processes.

```
y22cs139@rvrcse:~$ w
 12:49:10 up 14 days,  4:32, 98 users,  load average: 1.36, 2.00, 2.13
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
y22cs170 pts/0    10.1.3.39        11:11   11:42   0.10s  0.00s pager
t22csec4 pts/2    10.6.4.133       11:02    1:18   0.10s  0.09s -bash
y22cs140 pts/3    10.6.4.32        11:03    7:25   0.10s  0.09s -bash
cb182    pts/7    10.6.1.35        12:45    1:01   0.03s  0.02s -bash
y22cs197 pts/6    10.1.3.40        11:09    6.00s  0.09s  0.08s -bash
y22cs160 pts/14   10.6.4.31        11:11    5.00s  0.20s  0.19s -bash
cs152    pts/10   10.1.4.9         12:44    5.00s  0.06s  0.00s ./a.out
y22cs188 pts/8    10.6.4.1         11:09    1:34m  0.03s  0.02s -bash
y22cs159 pts/11   10.6.4.29        11:11    2:46   0.08s  0.07s -bash
```

13.***nl***: It is used to write each file to standard output with line numbers added.

```
y22cs139@rvrcse:~$ nl head1.txt
     1  aaa
     2  bbb
     3  iii
     4  ooo
     5  ppp
     6  yyy
     7  jjj
     8  ooo
     9  eee
y22cs139@rvrcse:~$
```

14.***grep(Global Regular Expression)***: It searches for a pattern in the given files.

a)grep –wnword filename: -wn used here to restrict matching to the whole words only with line numbers.

```
y22cs139@rvrcse:~$ cat grep1.txt
Well you know it's your bedtime,
So turn off the light,
Say all your prayers and thenWell you know its your bedtime,
So turn off the light,
Say all your prayers and then,
Oh you sleepy young heads dream of wonderful things,
Beautiful mermaids will swim through the sea,
And you will be swimming there too.
y22cs139@rvrcse:~$
```

```
y22cs139@rvrcse:~$ grep the grep1.txt
So turn off the light,
Say all your prayers and thenWell you know its your bedtime,
So turn off the light,
Say all your prayers and then,
Beautiful mermaids will swim through the sea,
And you will be swimming there too.
y22cs139@rvrcse:~$
```

b)grep –wnv word filename: This is used to display all the lines that doesnot have the word given along with line numbers.

```
y22cs139@rvrcse:~$ grep -wnv the grep1.txt
1:Well you know it's your bedtime,
3:Say all your prayers and thenWell you know its your bedtime,
5:Say all your prayers and then,
6:Oh you sleepy young heads dream of wonderful things,
8:And you will be swimming there too.
y22cs139@rvrcse:~$
```

15.***egrep(Extended Global Regular Expression)***: It is used to search for extended regular expression patterns.

a)egrep –E “starting letter.*ending letter” filename: It is used to find all the phrases that start with given starting letter and end with given ending letter.Here * indicates that display that phrase even if there were no letters between given starting letter and given ending letter.

If we use + then it displays all phrases only if they have atleast a single letter between the given starting letter and given ending letter.

```
y22cs139@rvrcse:~$ egrep -E "s.*w" grep1.txt
Say all your prayers and thenWell you know its your bedtime,
Oh you sleepy young heads dream of wonderful things,
Beautiful mermaids will swim through the sea,
And you will be swimming there too.
y22cs139@rvrcse:~$
```

```
y22cs139@rvrcse:~$ egrep -E "s.+w" grep1.txt
Say all your prayers and thenWell you know its your bedtime,
Oh you sleepy young heads dream of wonderful things,
Beautiful mermaids will swim through the sea,
y22cs139@rvrcse:~$
```

16.***fgrep(Fast or Fixed Global Regular Expression)***: It is used to search for fixed strings in specified files.

a)fgrep –F –n "stringname" filename: It gives output with line numbers.

```
y22cs139@rvrcse:~$ fgrep -F -n "ll" grep1.txt
1:Well you know it's your bedtime,
3:Say all your prayers and thenWell you know its your bedtime,
5:Say all your prayers and then,
7:Beautiful mermaids will swim through the sea,
8:And you will be swimming there too.
```

b)fgrep –F "stringname" filename: It gives output without line numbers.

```
y22cs139@rvrcse:~$ fgrep -F "ll" grep1.txt
Well you know it's your bedtime,
Say all your prayers and thenWell you know its your bedtime,
Say all your prayers and then,
Beautiful mermaids will swim through the sea,
And you will be swimming there too.
y22cs139@rvrcse:~$
```

17.***uniq(Unique)***: The "uniq" utility displays a file with all of its identical adjacent lines replaced by a single occurrence of the repeated line.

a)filter out duplicate adjacent lines and displays:

Syntax: uniq filename

```
y22cs139@rvrcse:~$ cat uniq1.txt
aaa bbb
aaa bbb
ccc ddd
eee fff
ccc ddd
ggg fff
yyy kkk
yyy kkk
mmm kkk
zzz qqq
zzz qqq
mmm kkk
```

```
y22cs139@rvrcse:~$ uniq uniq1.txt
aaa bbb
ccc ddd
eee fff
ccc ddd
ggg fff
yyy kkk
mmm kkk
zzz qqq
mmm kkk
```

b) displays a count with the lines:

Syntax: uniq –c filename

```
y22cs139@rvrcse:~$ uniq -c uniq1.txt
      2 aaa bbb
      1 ccc ddd
      1 eee fff
      1 ccc ddd
      1 ggg fff
      2 yyy kkk
      1 mmm kkk
      2 zzz qqq
      1 mmm kkk
y22cs139@rvrcse:~$
```

c) Ignore first field of each line:

Syntax: uniq -1 filename

```
y22cs139@rvrcse:~$ uniq -1 uniq1.txt
aaa bbb
ccc ddd
eee fff
ccc ddd
ggg fff
yyy kkk
zzz qqq
mmm kkk
y22cs139@rvrcse:~$
```

18.***chmod(Change Mode)***: It is used to change user's, group's and owner's read,write, and execute permissions of a file or directory.

**Use 'u' for User, 'g' for Group, 'o'for Owner.

** 'r' refers to Read permission, 'w' refers to Write permission, 'x' refers to Execute permission.

** Always in display it shows in the following order: User Group Owner

** Use ‘+’ to add permissions and use ‘–‘ to remove permissions.

a)Changing User permissions:

```
y22cs139@rvrcse:~$ chmod u-rw 1.txt
y22cs139@rvrcse:~$ ls -l
total 64
----rw-r-- 1 y22cs139 y22cs139   13 Feb  1 12:01 1.txt
-rw-rw-r-- 1 y22cs139 y22cs139 2409 Sep 15 12:59 ds.c
-rw-r--r-- 1 y22cs139 y22cs139 8980 Apr 16  2018 examples.desktop
-rw-rw-r-- 1 y22cs139 y22cs139  306 Feb  2 08:51 grep1.txt
-rw-rw-r-- 1 y22cs139 y22cs139   36 Feb  1 12:14 head1.txt
```

```
y22cs139@rvrcse:~$ chmod u+rw 1.txt
y22cs139@rvrcse:~$ ls -l
total 64
-rw-rw-r-- 1 y22cs139 y22cs139   13 Feb  1 12:01 1.txt
-rw-rw-r-- 1 y22cs139 y22cs139 2409 Sep 15 12:59 ds.c
-rw-r--r-- 1 y22cs139 y22cs139 8980 Apr 16  2018 examples.desktop
-rw-rw-r-- 1 y22cs139 y22cs139  306 Feb  2 08:51 grep1.txt
-rw-rw-r-- 1 y22cs139 y22cs139   36 Feb  1 12:14 head1.txt
```

19)***cmp(Compare)***: It is used to compare two files byte by byte and returns at which byte the files first differ.

```
y22cs139@rvrcse:~$ cat 1.txt
Hi hello RVR
y22cs139@rvrcse:~$ cat > 2.txt
Hi hell RVRJC
^C
y22cs139@rvrcse:~$ cmp 1.txt 2.txt
1.txt 2.txt differ: byte 8, line 1
y22cs139@rvrcse:~$
```

20.***diff(Difference)***: It is used to compare two files and displays a list of editing changes that would convert the first file into the second file. It displays three kinds of editing changes. They are:

‘a’ adding lines.

‘c’ changing lines.

‘d’ deleting lines.

```
y22cs139@rvrcse:~$ cat 1.txt
Hi hello RVR
y22cs139@rvrcse:~$ cat 2.txt
Hi hello RVR
y22cs139@rvrcse:~$ cat >> 2.txt
Eswar
^C
y22cs139@rvrcse:~$ diff 1.txt 2.txt
1a2
> Eswar
y22cs139@rvrcse:~$ diff 2.txt 1.txt
2d1
< Eswar
```

21.***paste***:It is used to parallel merge or join two files by outputting lines consisting of each line separated by a tab delimiter.It just prints but not change files contents.

```
y22cs139@rvrcse:~$ cat > paste1.txt
abc
abd
abe
abf
^C
y22cs139@rvrcse:~$ cat > paste2.txt
bca
bcb
bcd
bce
^C
y22cs139@rvrcse:~$ paste paste1.txt paste2.txt
abc     bca
abd     bcb
abe     bcd
abf     bce
```

22.***cut***:It is used for cutting out the sections from each line and displaying on standard output.It just prints but not change the original file.

i.Cutting as per bytes in each line: $cut –b byte1,byte2,byte3….. filename.txt

```
y22cs139@rvrcse:~$ cat grep1.txt
Well you know it's your bedtime,
So turn off the light,
Say all your prayers and thenWell you know its your bedtime,
So turn off the light,
Say all your prayers and then,
Oh you sleepy young heads dream of wonderful things,
Beautiful mermaids will swim through the sea,
And you will be swimming there too.
y22cs139@rvrcse:~$ cut -b 2,3,5 grep1.txt
el
o u
aya
o u
aya
h o
eat
ndy
y22cs139@rvrcse:~$
```

ii.Cutting based on range of bytes: $cut –b lowerbound-upperbound filename.txt

```
y22cs139@rvrcse:~$ cut -b 3-6 grep1.txt
ll y
 tur
y al
 tur
y al
 you
auti
d yo
```

iii.Cutting based on columns:

For column wise indexed printing use –c instead of –b in commands above.

iv.Cutting by using –f option which is used to print the specified field number and the –d option tells the compiler that what delimiter we used for seperating fields in our file.

$cut –d “delimiter used in file” –f field(or)wordnumber filename

```
y22cs139@rvrcse:~$ cut -d " " -f 1 grep1.txt
Well
So
Say
So
Say
Oh
Beautiful
And
```

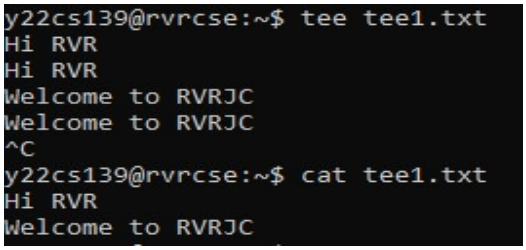
23.***join***: It is used to join the lines of two files based on common field. It just prints Upto common columns data in both the files involved. It ignores rows which are in extra.

```
y22cs139@rvrcse:~$ cat > join1.txt
1. Hi
2. Hello
3. Fine
4. RVR
^C
y22cs139@rvrcse:~$ cat > join2.txt
1. Welcome
2. Everyone
3. Nice
4. Done
^C
y22cs139@rvrcse:~$ join join1.txt join2.txt
1. Hi Welcome
2. Hello Everyone
3. Fine Nice
4. RVR Done
```

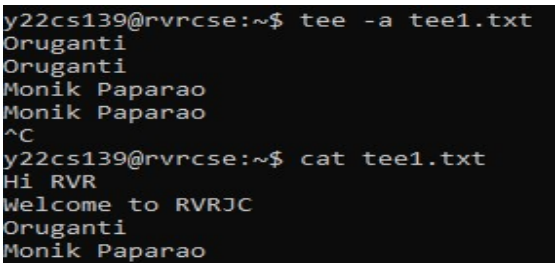
24.***tee***: It is used to read from standard input and write to standard output and files.

a.Without any options:

$tee filename: It is used to create a file by taking data input and writing it in that file and also prints the input automatically.



b. For appending data in exixting file: -a command is used to just append and print the standard input in standard output.



Module-3

Disk and Backup Utilities

1.***who***: It is used to print the information about the users who are currently logged in.

```
y22cs139@rvrcse:~$ who
y22cs3   pts/1        2024-02-29 09:42 (10.1.7.27)
y22cs48  pts/2        2024-02-29 10:00 (10.1.7.43)
y22cs20  pts/5        2024-02-29 10:00 (10.1.7.164)
y22cs87  pts/3        2024-02-29 09:58 (10.1.7.98)
y22cs59  pts/10       2024-02-29 10:02 (10.1.7.80)
123cs201 pts/6        2024-02-29 09:10 (10.1.7.90)
y22cs61  pts/11       2024-02-29 10:05 (10.1.7.1)
```

2.***du***: It is used to summarize disk usaqge of the set of files,recursively for directories.

```
y22cs139@rvrcse:~$ du
4       ./.cache
4       ./.gnupg/private-keys-v1.d
8       ./.gnupg
132     .
```

3.***df***: It displays the amount of diskspace available on file system containing each filename argument.

```
y22cs139@rvrcse:~$ df
Filesystem                         1K-blocks     Used Available Use% Mounted on
udev                                 8117400        0   8117400   0% /dev
tmpfs                                1630440     3880   1626560   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv  205314024 67747532 127064348  35% /
tmpfs                                8152196        0   8152196   0% /dev/shm
tmpfs                                   5120        4      5116   1% /run/lock
tmpfs                                8152196        0   8152196   0% /sys/fs/cgroup
/dev/sda2                             996780   160220    767748  18% /boot
/dev/sda1                             523248     6732    516516   2% /boot/efi
tmpfs                                1630436       16   1630420   1% /run/user/126
tmpfs                                1630436        0   1630436   0% /run/user/6342
```

4.***sed(Stream Editor)***: The utility sed scans one or more files and performs an editing action on all of the lines that match a particular condition.

a) Substituting text:(Example): Substuting first character with a space in each line.

i.Syntax:

$sed ‘s/^/character to be replaced at first of every line/’ filename

This doesnot change the content in the file but prints content just by inserting given character in the command.

```
y22cs139@rvrcse:~$ cat sed1.txt
Hi Hello RVR
Welcome to Guntur
Well.
y22cs139@rvrcse:~$ sed 's/^/@/' sed1.txt
@Hi Hello RVR
@Welcome to Guntur
@Well.
```

ii.Syntax:

$sed ‘s/^/ character to be replaced at first of every line’ file1 > file2

It copies the file1 content into file2. Content is copied just by adding given character given in command.It does not change content in file1.

```
y22cs139@rvrcse:~$  sed 's/^/@/' sed1.txt > sed2.txt
y22cs139@rvrcse:~$ cat sed2.txt
@Hi Hello RVR
@Welcome to Guntur
@Well.
y22cs139@rvrcse:~$ cat sed1.txt
Hi Hello RVR
Welcome to Guntur
Well.
```

*If file2 not exists initially it creates and copies content.

b)Deleting Text: Printing only by Deleting only those lines that contain the word given. It does not change the content in original file.

```
y22cs139@rvrcse:~$ cat sed1.txt
Hi Hello RVR
Welcome to Guntur
Well.
y22cs139@rvrcse:~$ sed '/Guntur/d' sed1.txt
Hi Hello RVR
Well.
```

Shell Programming:

Shell program provides the interface between user and an operating system when user locks in OS starts a shell for user.

The Bourne Shell is the original Unix Shell.
The Bourne Shell is used for scripting.

The Bourne Shell provides Command Based Programming to interpret and execute user Commands.

The types of Shell Programs are:

1.C-Shell

2.The Bourne Shell(sh)

3.The Korn Shell(ksh)

4.The Bourne Shell(bash)

Example Programs:

1.Program to Give a Prompt and read a text.

```
y22cs139@rvrcse:~$ echo "Enter your Number"
Enter your Number
y22cs139@rvrcse:~$ read a
Y22CS139
y22cs139@rvrcse:~$ echo "Number=$a"
Number=Y22CS139
```

2.Program to read two Numbers and find their sum.

```
y22cs139@rvrcse:~$  echo -n "Enter first Number:"
Enter first Number:y22cs139@rvrcse:~$  read Num1
35
y22cs139@rvrcse:~$ echo -n "Enter Second Number:"
Enter Second Number:y22cs139@rvrcse:~$ read Num2
78
y22cs139@rvrcse:~$ Sum=$(($Num1+$Num2))
y22cs139@rvrcse:~$ echo "Sum is $Sum"
Sum is 113
```

3.Program to read two numbers and perform Arithmetic Operations.

```
y22cs139@rvrcse:~$ cat ex3.sh
echo -n "Enter the first Number:"
read a
echo -n "Enter second Number:"
read b
c=$(($a+$b))
echo "Sum is $c"
c=$(($a-$b))
echo "Difference is $c"
c=$(($a*$b))
echo "Product is $c"
c=$(($a/$b))
echo "Quotient is $c"
c=$(($a%$b))
echo "Remainder is $c"
y22cs139@rvrcse:~$ bash ex3.sh
Enter the first Number:5
Enter second Number:3
Sum is 8
Difference is 2
Product is 15
Quotient is 1
Remainder is 2
```

4.Program to read two numbers and find the greatest number.(Using Simple if)

```
y22cs139@rvrcse:~$ cat ex4.sh
echo -n "Enter first number:"
read a
echo -n "Enter second number:"
read b
if [ $a -gt $b ]
then
        echo "Greatest Number is $a"
fi
y22cs139@rvrcse:~$ bash ex4.sh
Enter first number:54
Enter second number:32
Greatest Number is 54
```

5.Program to read two numbers and find the least number.(Using if else)

```
y22cs139@rvrcse:~$ cat ex5.sh
echo -n "Enter first Number:"
read a
echo -n "Enter second Number:"
read b
if [ $a -lt $b ]
then
        echo "Least Number is $a"
else
        echo "Least Number is $b"
fi
y22cs139@rvrcse:~$ bash ex5.sh
Enter first Number:43
Enter second Number:234
Least Number is 43
```

6.Program to read two numbers and find whether they are equal or not.

```
y22cs139@rvrcse:~$ cat ex6.sh
echo -n "Enter first number:"
read a
echo -n "Enter econd number:"
read b
if [ $a == $b ]
then
        echo "Given Numbers are Equal"
else
        echo "Given Numbers are not Equal"
fi
y22cs139@rvrcse:~$ bash -f ex6.sh
Enter first number:5
Enter econd number:2
Given Numbers are not Equal
```

7. Program to read a number and find its type either positive or negative or neither both.

```
y22cs139@rvrcse:~$ cat ex7.sh
echo -n "Enter a Number:"
read a
if [ $a -lt 0 ]
then
        echo "Given Number is Negative"
elif [ $a -gt 0 ]
then
        echo "Given Number is Positive"
else
        echo "Given Number is Neither Positive Nor Negative"
fi
y22cs139@rvrcse:~$ bash -f ex7.sh
Enter a Number:5
Given Number is Positive
```

8.Program to find given character is a consonant or vowel.

```
y22cs139@rvrcse:~$ cat ex8.sh
echo -n "Enter a Character:"
read char
case $char in
        a) echo "Entered character $char is a Vowel";;
        e) echo "Entered character $char is a Vowel";;
        i) echo "Entered character $char is a Vowel";;
        o) echo "Entered character $char is a Vowel";;
        u) echo "Entered character $char is a Vowel";;
        A) echo "Entered character $char is a Vowel";;
        E) echo "Entered character $char is a Vowel";;
        I) echo "Entered character $char is a Vowel";;
        O) echo "Entered character $char is a Vowel";;
        U) echo "Entered character $char is a Vowel";;
        *) echo "Entered character $char is a Consonant";;
esac
y22cs139@rvrcse:~$ bash -f ex8.sh
Enter a Character:e
Entered character e is a Vowel
y22cs139@rvrcse:~$
```

9.Write a Program to print first five Whole Numbers using While Loop.

```
y22cs139@rvrcse:~$ cat ex9.sh
count=0
while [ $count -lt 5 ]
do
        echo "Loop Count is $count"
        count=$((count+1))
done

y22cs139@rvrcse:~$ bash -f ex9.sh
Loop Count is 0
Loop Count is 1
Loop Count is 2
Loop Count is 3
Loop Count is 4
y22cs139@rvrcse:~$
```

10.Write a Program to read two numbers and find their sum.

```
y22cs139@rvrcse:~$ cat ex10.sh
echo "Given First Number is:"$1
echo "Given Second Number is:"$2
c=`expr $1 + $2`
echo "Sum is $c"
y22cs139@rvrcse:~$ bash -f ex10.sh 20 30
Given First Number is:20
Given Second Number is:30
Sum is 50
```

Module-4

Programmable Text Processing

awk is a programmable text-processing utility that scans the lines of its input and performs actions on every line that matches a particular criterion

a) Accessing individual files:

```
y22cs139@rvrcse:~$ cat awk1.txt
Wish I was floating in blue across the sky,
My imagination is strong,
And I often visit the days
When everything seemed so clear.
Now I wonder what I'm doing here at all...
y22cs139@rvrcse:~$ awk '{ print NF, $0 }' awk1.txt
9 Wish I was floating in blue across the sky,
4 My imagination is strong,
6 And I often visit the days
5 When everything seemed so clear.
9 Now I wonder what I'm doing here at all...
```

b) Begin and End:

```
y22cs139@rvrcse:~$ cat awk2.txt
BEGIN { print "Start of file:", FILENAME }
{ print $1 $3 $NF }
END { print "End of file" }
y22cs139@rvrcse:~$ awk -f awk2.txt awk1.txt
Start of file:
Wishwassky,
Myisstrong,
Andoftendays
Whenseemedclear.
Nowwonderall...
End of file
```

c) Operators:

```
y22cs139@rvrcse:~$ cat awk3.txt
NR > 1 && NR < 4 { print NR, $1, $3, $NF }
y22cs139@rvrcse:~$ awk -f awk3.txt awk1.txt
2 My is strong,
3 And often days
```

d) Variables:

```
y22cs139@rvrcse:~$ cat awk4.txt
BEGIN { print "Scanning file" }
{
printf "line %d: %s\n", NR, $0;
lineCount++;
wordCount += NF;
}
END { printf "lines = %d, words = %d\n", lineCount, wordCount }
y22cs139@rvrcse:~$ awk -f awk4.txt awk1.txt
Scanning file
line 1: Wish I was floating in blue across the sky,
line 2: My imagination is strong,
line 3: And I often visit the days
line 4: When everything seemed so clear.
line 5: Now I wonder what I'm doing here at all...
lines = 5, words = 33
```

e) Control Structures:

```
y22cs139@rvrcse:~$ cat awk5.txt
{
for (i = NF; i >= 1; i--)
printf "%s ", $i;
printf "\n";
}
y22cs139@rvrcse:~$ awk -f awk5.txt awk1.txt
sky, the across blue in floating was I Wish
strong, is imagination My
days the visit often I And
clear. so seemed everything When
all... at here doing I'm what wonder I Now
```

f) Extended Regular Expressions:

```
y22cs139@rvrcse:~$ cat awk6.txt
/t.*e/ { print $0 }
y22cs139@rvrcse:~$ awk -f awk6.txt awk1.txt
Wish I was floating in blue across the sky,
And I often visit the days
When everything seemed so clear.
Now I wonder what I'm doing here at all...
```

g) Condition Ranges:

```
y22cs139@rvrcse:~$ cat awk7.txt
/strong/ , /clear/ { print $0 }
y22cs139@rvrcse:~$ awk -f awk7.txt awk1.txt
My imagination is strong,
And I often visit the days
When everything seemed so clear.
```

h) Field-Separators:

```
y22cs139@rvrcse:~$ cat awk3.txt
NR > 1 && NR < 4 { print NR, $1, $3, $NF }
y22cs139@rvrcse:~$ cat awk8.txt
Wish:I:was:floating:in:blue:across:the:sky,
My:imagination:is:strong,
And:I:often:visit:the:days
When:everything:seemed:so:clear.
Now:I:wonder:what:I'm:doing:here:at:all...
y22cs139@rvrcse:~$ awk -F: -f awk3.txt awk8.txt
2 My is strong,
3 And often days
```

i) Built-in –Functions:

```
y22cs139@rvrcse:~$ cat awk9.txt
1.1 a
2.2 at
3.3 eat
4.4 beat
y22cs139@rvrcse:~$ cat awk10.txt
{
printf "$1 = %g ", $1;
printf "exp = %.2g ", exp ($1);
printf "log = %.2g ", log ($1);
printf "sqrt = %.2g ", sqrt ($1);
printf "int = %d ", int ($1);
printf "substr (%s, 1, 2) = %s\n", $2, substr($2, 1, 2);
}
y22cs139@rvrcse:~$ awk -f awk10.txt awk9.txt
$1 = 1.1 exp = 3 log = 0.095 sqrt = 1 int = 1 substr (a, 1, 2) = a
$1 = 2.2 exp = 9 log = 0.79 sqrt = 1.5 int = 2 substr (at, 1, 2) = at
$1 = 3.3 exp = 27 log = 1.2 sqrt = 1.8 int = 3 substr (eat, 1, 2) = ea
$1 = 4.4 exp = 81 log = 1.5 sqrt = 2.1 int = 4 substr (beat, 1, 2) = be
```

Module-5

Shell Scripting

1) Write a shell script program for the following.

a) To create a directory and list all the directory files in a directory.

```
y22cs139@rvrcse:~$ cat s1.sh
echo "Enter the directory name to create:"
read dir
if [ -e $dir ]:
then
        echo "Directory already existed"
else
        mkdir $dir
        echo "Directory Created!"
fi
echo "enter the directory that already existed"
read dr
if [ -e $dr ]:
then
         cd $dr
         ls
         cd ..

else
        echo "Directory not existed"
fi

y22cs139@rvrcse:~$ sh s1.sh
Enter the directory name to create:
linux
Directory Created!
enter the directory that already existed
ompr
ompr1.txt  ompr2.txt  omprtest  omprtest1
y22cs139@rvrcse:~$
```

b) To display a list of all the files in the current directory

```
y22cs139@rvrcse:~$ cat s2.sh
echo enter the directory name:
read dir
if [ $dir ]:
then
        ls $dir
else
        echo directory not valid!!
fi
y22cs139@rvrcse:~$ sh s2.sh
enter the directory name:
ompr
ompr1.txt  ompr2.txt  omprtest  omprtest1
```

c) To count no of lines, words, and characters of an input file.

```
y22cs139@rvrcse:~$ cat s3.sh
echo "Enter the file name:"
read fname
if [ -e $fname ]:
then
echo Word count is:
wc -w $fname
echo line count is:
wc -l $fname
echo char count is:
wc -m $fname
else
        echo file not existed
fi

y22cs139@rvrcse:~$ sh s3.sh
Enter the file name:
ompr.txt
Word count is:
8 ompr.txt
line count is:
4 ompr.txt
char count is:
55 ompr.txt
```

d) To accept a file name starting and ending line numbers as arguments and display all the lines between given line numbers.

```
y22cs139@rvrcse:~$ cat s4.sh
echo -n "Enter a file name:"
read filename
echo -n "Enter staring line number:"
read s
echo -n "Enter ending linbe number:"
read e
sed -n $s,$e\p $filename
y22cs139@rvrcse:~$ sh s4.sh
Enter a file name:ompr.txt
Enter staring line number:2
Enter ending linbe number:4
Guntur,Andhra Pradesh.
welcome hi
linux
```

e) To deletes all lines containing the specified word in one or more files supplied as arguments to it.

```
y22cs139@rvrcse:~$ cat s5.sh
echo -n "Enter the word to delete: "
read word
for i in $@
do
        sed '/'$word'/d' $i
done

y22cs139@rvrcse:~$ sh s5.sh s1.sh
Enter the word to delete: echo
read dir
if [ -e $dir ]:
then
else
        mkdir $dir
fi
read dr
if [ -e $dr ]:
then
         cd $dr
         ls
         cd ..

else
fi
```

f) To test whether the given file is existing or not.

```
y22cs139@rvrcse:~$ cat s6.sh
echo -n "Enter file name:"
read fname
if [ -e $fname ]
then
        echo "File "$fname" exists"
else
        echo "File "$fname" does not exists"
fi
y22cs139@rvrcse:~$ sh s6.sh
Enter file name:ompr1050.txt
File ompr1050.txt does not exists
y22cs139@rvrcse:~$ sh s6.sh
Enter file name:ompr.txt
File ompr.txt exists
y22cs139@rvrcse:~$
```

g) To read, delete and append a file.

```
y22cs139@rvrcse:~$ cat s7.sh
echo -n "Enter the filename to read:"
read fname
if [ -e $fname ]
then
        echo "Contents of the file:"
        cat $fname
else
        echo "File doesnot exists"
fi
echo -n "Enter filename to delete:"
read fname
if [ -e $fname ]
then
        rm $fname
        echo "File Deleted"
else
        echo "File doesnot exists"
fi
echo  -n "Enter filename to append:"
read fname
if [ -e $fname ]
then
        echo "Enter the text to append:"
        cat >> $fname
else
        echo "File doesnot exists"
fi
y22cs139@rvrcse:~$ sh s7.sh
Enter the filename to read:sort1.txt
Contents of the file:
999
656
932
65
1
Enter filename to delete:sort2.txt
File Deleted
Enter filename to append:sort1.txt
Enter the text to append:
2004
^C
```

h) To store all command line arguments to an array and print.

```
y22cs139@rvrcse:~$ sh s8.sh abcd efgh
The arguments are:
abcd
efgh
y22cs139@rvrcse:~$ cat s8.sh
arr=$@
echo "The arguments are:"
for i in $arr
do
        echo $i
done
y22cs139@rvrcse:~$ sh s8.sh abcd efgh
The arguments are:
abcd
efgh
```

i) To print the calendar month by default.

```
y22cs139@rvrcse:~$ cat s9.sh
mon=$1
yr=$2
echo "Calender of given date: "
cal $mon $yr
y22cs139@rvrcse:~$ sh s9.sh
Calender of given date:
     April 2024
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

MODULE-6

FILE MANAGEMENT SYSTEM CALLS

1. Write a program on File management System Calls: open (), read (), write (), close ().

```
y22cs139@rvrcse:~$ cat f1.c
#include<stdio.h>
#include<fcntl.h>
#include<stdlib.h>
#include<string.h>
int main()
{
        int fd1,fd2;
        fd1=open("ompr.txt",O_RDONLY);
        fd2=open("test1.txt",O_CREAT|O_RDWR,0700);
        printf("fd1=%d\n",fd1);
        printf("fd2=%d\n",fd2);
        char *c=(char*)malloc(20*sizeof(char));
        int s=read(fd1,c,10);
        c[s]='\0';
        printf("Contents of the first %d bytes of fd1:%s\n",s,c);
        write(fd2,"RVRJCCE,RVJC,Guntur\n",19);
        close(fd1);
        close(fd2);
}
y22cs139@rvrcse:~$ ./a.out
fd1=3
fd2=4
Contents of the first 10 bytes of fd1:Welcome to
y22cs139@rvrcse:~$ cat test1.txt
RVRJCCE,RVJC,Guntury22cs139@rvrcse:~$
```

2. Write a program on File handling system call: perror ().

```
y22cs139@rvrcse:~$ cat f2.c
#include<stdio.h>
#include<fcntl.h>
#include<sys/file.h>
#include<errno.h>
void main()
{
        int fd1,fd2;
        fd1=open("nonexist.txt",O_RDONLY);
        if(fd1==-1)
        {
                printf("errno=%d\n",errno);
                perror("Couldn't open the fiel to read");
        }
        fd2=open("nonexistent.txt",O_WRONLY);
        if(fd2==-1)
        {
                printf("errno=%d\n",errno);
                perror("Colud'nt open the file to write\n");
        }
}
y22cs139@rvrcse:~$ ./a.out
errno=2
Couldn't open the fiel to read: No such file or directory
errno=2
Colud'nt open the file to write
: No such file or directory
```

3. Write a program for demonstrating dup () and dup2() system calls.

```
y22cs139@rvrcse:~$ cat f3.c
#include<stdio.h>
#include<fcntl.h>
#include<unistd.h>
void main()
{
        int old,new;
        old=open("ompr.txt",O_RDWR);
        printf("File descriptor is:%d\n",old);
        new=dup(old);
        printf("New File descriptor is:%d\n",new);
        close(old);
        close(new);
}
y22cs139@rvrcse:~$ ./a.out
File descriptor is:3
New File descriptor is:4
```

```
y22cs139@rvrcse:~$ cat f3_2.c
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
void main()
{
        int old,new;
        old=open("test1.txt",O_RDWR);
        printf("File descriptor is %d\n",old);
        new=dup2(old,7);
        printf("New file descriptor is %d\n",new);
}
y22cs139@rvrcse:~$ ./a.out
File descriptor is 3
New file descriptor is 7
```

MODULE – 7

PROCESS MANAGEMENT SYSTEM CALLS

1. Write a program to create two processes, to run a loop in which one process adds all even numbers and other process adds all odd numbers (use fork() system call).

```
y22cs139@rvrcse:~$ cat sc1.c
#include<stdio.h>
#include<sys/types.h>
void EvenSum()
{
        int sum=0;
        for(int i=2;i<=10;i+=2)
                sum=sum+i;
        printf("\nSum of Even Numbers: %d\n\n",sum);
}
void OddSum()
{
        int sum=0;
        for(int i=1;i<=10;i+=2)
                sum+=i;
        printf("\nSum of Even Numbers: %d\n\n",sum);
}
int main()
{
        pid_t pid=fork();
        if(pid==0)
                EvenSum();
        else
                OddSum();
}
y22cs139@rvrcse:~$ ./a.out

Sum of Even Numbers: 25

y22cs139@rvrcse:~$
Sum of Even Numbers: 30
```

2. Write a Program to create orphan process.

```
y22cs139@rvrcse:~$ cc orphan.c
y22cs139@rvrcse:~$ cat orphan.c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
        int pid=fork();
        if(pid>0)
                printf("In Parent Process\n");
        else if(pid==0)
        {
                sleep(10);
                printf("In Child Process\n");
        }
}
y22cs139@rvrcse:~$ ./a.out
In Parent Process
y22cs139@rvrcse:~$ In Child Process
```

3) Write a Program to create a zombie process and how to avoid Zombie using wait().

```
y22cs139@rvrcse:~$ cat sc2.c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
        pid_t child=fork();
        if(child>0)
        {
                printf("Parent Process Start\n");
                wait(&child);
                printf("Parent Process ends\n");
        }
        else if(child==0)
        {
                printf("Child Process Start\n");
                sleep(10);
                printf("Child Process End\n");
                exit(0);
        }
}
y22cs139@rvrcse:~$ ./a.out
Parent Process Start
Child Process Start
Child Process End
Parent Process ends
```

MODULE – 8

OPERATIONS ON SIGNALS

1) Write a program for Requesting an alarm signal to execute user defined alarm handler.

```
y22cs139@rvrcse:~$ cat alarm.c
#include<stdio.h>
#include<signal.h>
int alarmFlag=0;
void alarmHandler()
{
        printf("An Alarm Clock Signal was received\n");
        alarmFlag=1;
}
void main()
{
        signal(SIGALRM,alarmHandler);
        alarm(5);
        printf("Looping.....\n");
        while(!alarmFlag)
        {
                printf("Inside the Loop\n");
                pause();
        }
        printf("Loop ends due to alarm signal\n");
}
y22cs139@rvrcse:~$ ./a.out
Looping.....
Inside the Loop
An Alarm Clock Signal was received
Loop ends due to alarm signal
```

2) Write a program to demonstrate Suspending and Resuming Processes.

```
y22cs139@rvrcse:~$ cat processes.c
#include<stdio.h>
#include<signal.h>
void main()
{
        int pid1,pid2;
        pid1=fork();
        if(pid1==0)
        {
                while(1)
                {
                        printf("Process-1 is Alive\n");
                        sleep(1);
                }
        }
        pid2=fork();
        if(pid2==0)
        {
                while(1)
                {
                        printf("Process-2 is Alive\n");
                        sleep(1);
                }
        }
        sleep(3);
        kill(pid1,SIGSTOP);
        sleep(3);
        kill(pid1,SIGCONT);
        sleep(3);
        kill(pid1,SIGINT);
        kill(pid2,SIGINT);
}
y22cs139@rvrcse:~$ ./a.out
Process-1 is Alive
Process-2 is Alive
Process-1 is Alive
Process-2 is Alive
Process-1 is Alive
Process-2 is Alive
Process-2 is Alive
Process-2 is Alive
Process-2 is Alive
Process-1 is Alive
Process-2 is Alive
Process-1 is Alive
Process-2 is Alive
Process-1 is Alive
Process-2 is Alive
```

MODULE – 9

Inter Process Communication

1. Write a program to implement the concept of unnamed pipes.

```
y22cs139@rvrcse:~$ cat piping.c
#include<stdio.h>
#include<string.h>
#define READ 0
#define WRITE 1
char *phrase="This is a Piped Text!!!";
void main()
{
        int fd[2],bytesRead;
        char message[100];
        pipe(fd);
        if(fork()==0)
        {
                close(fd[READ]);
                write(fd[WRITE],phrase,strlen(phrase)+1);
                close(fd[WRITE]);
        }
        else
        {
                close(fd[WRITE]);
                bytesRead=read(fd[READ],message,100);
                printf("Read %d bytes: %s\n",bytesRead,message);
                close(fd[READ]);
        }

}
y22cs139@rvrcse:~$ ./a.out
Read 24 bytes: This is a Piped Text!!!
```

2) Write a program to implement the concept of named pipes.

```
y22cs139@rvrcse:~$ mknod myPipe p
y22cs139@rvrcse:~$ ls -l myPipe
prw-rw-r-- 1 y22cs139 y22cs139 0 Apr 10 09:14 myPipe
y22cs139@rvrcse:~$ chmod ug-rw myPipe
y22cs139@rvrcse:~$ ls -l myPipe
p------r-- 1 y22cs139 y22cs139 0 Apr 10 09:14 myPipe
```