IT343 – Operating System (5th Sem)

Lab Manual

Practical - 1

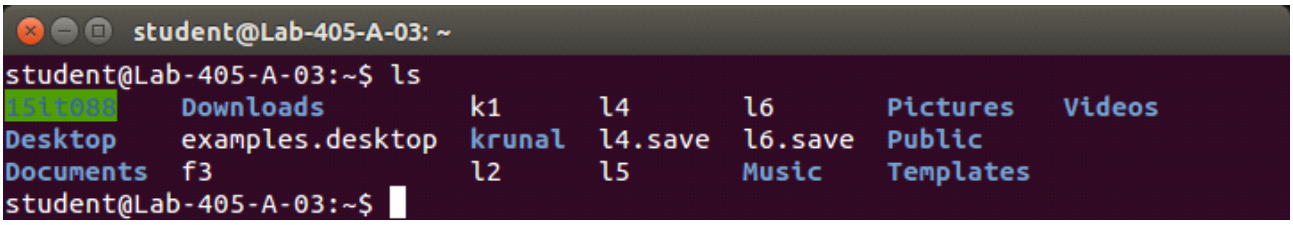
Implement the basic and advanced Linux commands.

a. Linux File System Overview.
b. Linux Utility Commands.
- General Commands: telnet, login ,man , logname, uname, who, who am I , tty, date, cal ,echo ,expr ,bc
- File Commands : mkdir, cd ,cd ..,pwd, rm ,cp ,mv ,cat ,touch, ls , ln
- Filter Commands: head ,tail ,cut ,paste,sort ,unique ,tr ,grep ,cmp

c. Manage Access control for the Users and Group
- Chmod ,chown ,umask , ls –l , addgroup ,adduer, passwd ,inode

d. Special commands
- arch, dmesg, uptime, id, last, finger, top, w, time sleep, history

Command: ls
Command description: listing of data and files in current directory
syntax: ls
Output:



Command:ls -l
Command description:long listing of files (detailed information obout files)
syntax:ls -l
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ ls -l
total 48
drwxrwxrwx 2 student student 4096 Jan  1  2008 15it088
drwxr-xr-x 3 student student 4096 Jan  1  2008 Desktop
drwxr-xr-x 2 student student 4096 Jan  1 00:02 Documents
drwxr-xr-x 2 student student 4096 Jan  1 00:02 Downloads
-rw-r--r-- 1 student student 8980 Jan  1  2008 examples.desktop
-rw-rw-r-- 1 student student    0 Jan  1 00:02 f3
drwxr-xr-x 2 student student 4096 Jan  1 00:02 Music
drwxr-xr-x 2 student student 4096 Jul 10  2017 Pictures
drwxr-xr-x 2 student student 4096 Jan  1 00:02 Public
drwxr-xr-x 2 student student 4096 Jan  1 00:02 Templates
drwxr-xr-x 3 student student 4096 Jan  1 00:10 Videos
student@Lab-405-A-03:~$
```

Command: bc
Command description: basic calculator
syntax: 2+2 , 2*2 , 3-2 , 4/2
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ bc
bc 1.06.95
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
2+2
4
2*5
10
```

Command:cal
Command description:calender of current month
syntax:cal
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ cal
    January 2008
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 31

student@Lab-405-A-03:~$
```

Command: cd
Command description: change directory
syntax: cd
Output:

```
student@Lab-405-A-03: ~/Music
student@Lab-405-A-03:~$ cd Music
student@Lab-405-A-03:~/Music$
```

Command: cd..
Command description: for come to your parent directory
syntax:cd..
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~/Music$ cd ..
student@Lab-405-A-03:~$
```

Command: date
Command description: current date
syntax:date
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ date
Tue Jan  1 00:40:34 IST 2008
student@Lab-405-A-03:~$
```

Command: echo
Command description: print message on screen
syntax: echo message
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ echo hello word
hello word
student@Lab-405-A-03:~$
```

Command: expr
Command description: for any arithmatic and logical operation
syntax: expr a + b , expr a * b
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ expr 2 + 4
6
student@Lab-405-A-03:~$ expr 3 \* 5
15
student@Lab-405-A-03:~$ expr 4 / 2
2
student@Lab-405-A-03:~$ expr 5 - 6
-1
student@Lab-405-A-03:~$
```

Command: grep
Command description: it is use to search pattern / word from file
syntax: grep pattern filename
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ grep a l6
are
student@Lab-405-A-03:~$
```

Command: head , tail
Command description: for first and last lines
syntax: head filename (bydefault first 10 lines) , tail filename (bydefault last 10 lines) , head -2 filename , tail -2 filename
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ head -2 l6
hii
how
student@Lab-405-A-03:~$ tail -2 l6
are
you
student@Lab-405-A-03:~$
```

Command: logname
Command description: login information
syntax: logname
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ logname
student
student@Lab-405-A-03:~$
```

Command: mkdir
Command description: create new directory
syntax: mkdir directoryname
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ mkdir krunal
student@Lab-405-A-03:~$ ls
15it038  Documents  examples.desktop  krunal  Pictures  Templates
Desktop  Downloads  f3                Music   Public    Videos
```

Command: pwd
Command description: present working directory
syntax:pwd
Output:

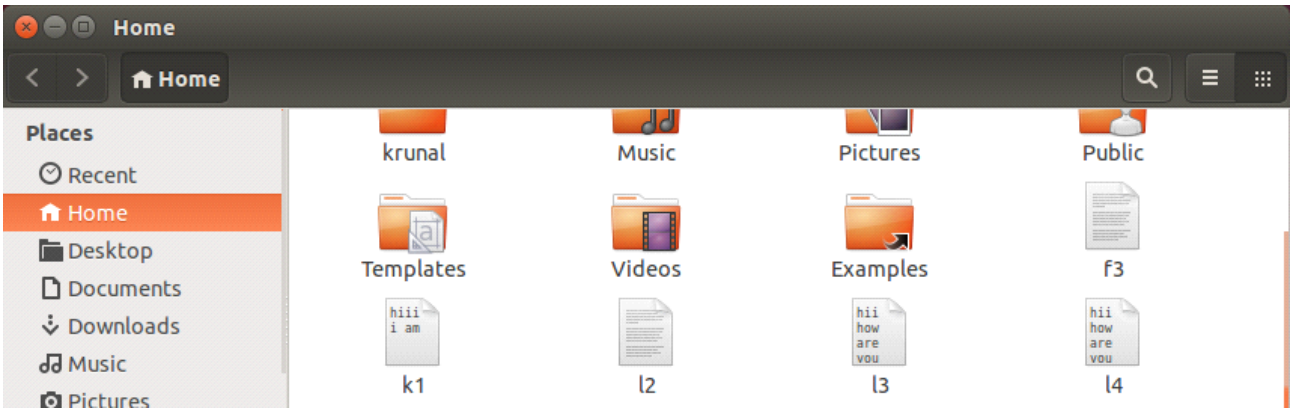
```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ pwd
/home/student
student@Lab-405-A-03:~$
```

Command: sort
Command description: sort file content
syntax: sort filename
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ sort l6
are
hii
how
you
student@Lab-405-A-03:~$
```

Command: touch
Command description: create any number of files
syntax: touch filename1 filename2 filename3 filename4 ...
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ touch l2 l3 l5
student@Lab-405-A-03:~$ touch -c l3
student@Lab-405-A-03:~$ vi l3
student@Lab-405-A-03:~$
```



Command: tty
Command description: terminal information
syntax: tty
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ tty
/dev/pts/0
student@Lab-405-A-03:~$
```

Command: uname
Command description: information about system
syntax: uname
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ uname
Linux
student@Lab-405-A-03:~$
```

Command: who
Command description: give all the working users in the system
syntax: who
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ who
student  :0           2008-01-01 00:01 (:0)
student  pts/0        2008-01-01 00:16 (:0)
student@Lab-405-A-03:~$
```

Command: who am i
Command description: give name of that user which is currently logged in.
syntax: who am i
Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ who am i
student  pts/0        2008-01-01 00:16 (:0)
student@Lab-405-A-03:~$
```

Command: chmod
Command description: this command is use to change the permission of files.
syntax: chmod rwxrwxrwx filename
Output:

```
-rw-rw-r-- 1 student student   34 Jan  1  2008 l4.save
-rw-rw-r-- 1 student student    0 Jan  1  2008 l5
-rw-rw-r-- 1 student student   18 Jan  1  2008 l6
```

```
student@Lab-405-A-03:~$ chmod 754 l5
```

```
-rwxr-xr-- 1 student student    0 Jan  1  2008 l5
```

Command: compgen
Command description: list existing users and groups in the system.

syntax: compgen -u (users) , compgen -g (groups)
Output:

```
student@Lab-405-A-03:~$ compgen -g
root
daemon
bin
sys
adm
tty
disk
lp
mail
news
uucp
man
```

```
student@Lab-405-A-03:~$ compgen -u
root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
```

Command: sudo su , addgroup
Command description: move to the root directory
syntax: sudo su , addgroup groupname
Output:

```
student@Lab-405-A-03:~$ sudo su
[sudo] password for student:
root@Lab-405-A-03:/home/student# addgroup g3
Adding group `g3' (GID 1002) ...
Done.
root@Lab-405-A-03:/home/student# compgen -g
root
daemon
bin
sys
```

```
m
g3
```

Command: chown
Command description: to change the owner of the group

syntax: chown username filename
Output:

```
root@Lab-405-A-03:/home/student# adduser u1
Adding user `u1' ...
Adding new group `u1' (1003) ...
Adding new user `u1' (1002) with group `u1' ...
Creating home directory `/home/u1' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for u1
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
root@Lab-405-A-03:/home/student# compgen -u
root
daemon
bin
```

```
u1
```

```
-rwxr-xr-- 1 student student    0 Jan  1  2008 l5
```

```
-rwxr-xr-- 1 u1      student    0 Jan  1  2008 l5
```

Command: history
Command description: command which executed previous
syntax: history
Output:

```
root@Lab-405-A-03:/home/student# history
    1  addgroup g3
    2  compgen -g
    3  compgen -u
    4  adduser u1
    5  compgen -u
    6  ls -l
    7  chown u1 l5
    8  ls -l
    9  clear
   10  history
```

Command: ctrl+d
Command description: to move out from root
syntax: ctrl+d

Output:

```
root@Lab-405-A-03:/home/student# exit
student@Lab-405-A-03:~$
```

Command: arch
Command description: architechre of current host
syntax: arch
Output:

```
student@Lab-405-A-03:~$ arch
i686
```

Command: w
Command description: information about users (login)
syntax: w
Output:

```
student@Lab-405-A-03:~$ w
 00:44:17 up 42 min,  2 users,  load average: 0.57, 0.32, 0.22
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
student  :0       :0               00:02   ?xdm?   4:27   0.23s init --user
student  pts/0    :0               00:27    1.00s  0.06s  0.00s w
student@Lab-405-A-03:~$
```

Command: ls -il , ls -il f1
Command description: give inod number
syntax: ls -il , ls -il filename
Output:

```
student@Lab-405-A-03:~$ ls -il
total 72
394522 drwxrwxrwx 2 student student 4096 Jan  1  2008 15it088
393993 drwxr-xr-x 3 student student 4096 Jan  1 00:07 Desktop
393997 drwxr-xr-x 2 student student 4096 Jan  1 00:02 Documents
393994 drwxr-xr-x 2 student student 4096 Jan  1 00:02 Downloads
393988 -rw-r--r-- 1 student student 8980 Jan  1  2008 examples.desktop
394372 -rw-rw-r-- 1 student student    0 Jan  1 00:02 f3
394560 -rw-rw-r-- 1 student student   21 Jan  1  2008 k1
394566 drwxrwxr-x 2 student student 4096 Jan  1  2008 krunal
```

```
student@Lab-405-A-03:~$ ls -il l4
394593 -rw-rw-r-- 1 student student 18 Jan  1  2008 l4
```

Command: umask
Command description: filesystem creation mask
syntax: umask
Output:

```
root@Lab-405-A-03:/home/student# umask
0022
```

Command: ps
Command description: files with process-id
syntax: ps
Output:

```
root@Lab-405-A-03:/home/student# ps
  PID TTY          TIME CMD
 4175 pts/0    00:00:00 sudo
 4176 pts/0    00:00:00 su
 4177 pts/0    00:00:00 bash
 4403 pts/0    00:00:00 ps
root@Lab-405-A-03:/home/student#
```

Command: last
Command description: listing of most recently login users.
syntax: last
Output:

```
root@Lab-405-A-03:/home/student# last
student  pts/0        :0               Tue Jan  1 00:27   still logged in
student  pts/0        :0               Tue Jan  1 00:12 - 00:27  (00:15)
student  pts/0        :0               Tue Jan  1 00:07 - 00:12  (00:04)
student  :0           :0               Tue Jan  1 00:02   still logged in
reboot   system boot  3.16.0-30-generi Tue Jan  1 00:01 - 00:54  (00:52)
student  pts/0        :0               Tue Jan  1 01:15 - 01:15  (00:00)
student  pts/0        :0               Tue Jan  1 00:16 - 01:13  (00:56)
student  pts/12       :0               Tue Jan  1 00:02 - 00:16  (00:14)
student  :0           :0               Tue Jan  1 00:01 - down   (01:56)
reboot   system boot  3.16.0-30-generi Tue Jan  1 00:01 - 01:57  (01:56)
student  pts/23       :0               Tue Jan  1 00:52 - crash  (00:-51)
```

Command: uptime
Command description: how long the system has been running.
syntax: uptime
Output:

```
root@Lab-405-A-03:/home/student# uptime
 00:56:12 up 54 min,  2 users,  load average: 0.16, 0.26, 0.23
```

Command: top
Command description: dynamic realtime view of running time. It can display of system summary information. As well as list of process and threads currently being managed by kernel.
syntax: top -hv
Output:

```
root@Lab-405-A-03:/home/student# top -hv
  procps-ng version 3.3.9
Usage:
  top -hv | -bcHiOSs -d secs -n max -u|U user -p pid(s) -o field -w [cols]
```

Command: dmesg
Command description: it will examines and control kernel ring buffer.
syntax: dmesg

Command: id
Command description: print real and effective users and group id.
syntax: id
Output:

```
root@Lab-405-A-03:/home/student# id
uid=0(root) gid=0(root) groups=0(root)
```

Command: ln
Command description: create link between two files.
syntax: ln
Output:

```
student@Lab-405-A-03:~$ ln -s l3 k6
student@Lab-405-A-03:~$ cat k6
hi
i am
student@Lab-405-A-03:~$ gedit l3
student@Lab-405-A-03:~$ cat l3
hi
i am krunal
study in charusat.
student@Lab-405-A-03:~$ cat k6
hi
i am krunal
study in charusat.
```

Command: cal july 2017
Command description: calender of july 2017
syntax: cal july 2017
Output:

```
     July 2017
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 31
```

Command: cal 2017
Command description: calender of 2017
syntax: cal 2017
Output:

```
       April                   May                    June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                   1      1  2  3  4  5  6                 1  2  3
 2  3  4  5  6  7  8   7  8  9 10 11 12 13   4  5  6  7  8  9 10
 9 10 11 12 13 14 15  14 15 16 17 18 19 20  11 12 13 14 15 16 17
16 17 18 19 20 21 22  21 22 23 24 25 26 27  18 19 20 21 22 23 24
23 24 25 26 27 28 29  28 29 30 31           25 26 27 28 29 30
30

        July                  August                September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                   1         1  2  3  4  5                    1  2
 2  3  4  5  6  7  8   6  7  8  9 10 11 12   3  4  5  6  7  8  9
 9 10 11 12 13 14 15  13 14 15 16 17 18 19  10 11 12 13 14 15 16
16 17 18 19 20 21 22  20 21 22 23 24 25 26  17 18 19 20 21 22 23
23 24 25 26 27 28 29  27 28 29 30 31        24 25 26 27 28 29 30
30 31
```

Command: cal -m (1,2,....,10)
Command description:
Syntax: cal -m 2
Output:

```
    January 2008
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 31
```

Command: cal -h
Command description: Turns off highlighting of today.
Syntax: cal -h
Output:

```
    January 2008
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 31
```

Command: bc -v
Command description: Print the version number and copyright and quit.
Syntax: bc -v
Output:

```
bc 1.06.95
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.
```

Command: ls -a
Command description: Print the version number and copyright and quit.
syntax:ls -a
Output:

```
student@Lab-405-A-03:~$ ls -a
.               .compiz          .gconf          l4.save    .profile
..              .config          .ICEauthority   l5         Public
15it088         Desktop          k1              l6         Templates
15it88          .dmrc            k6              l6.save    Videos
.bash_history   Documents        krunal          .local     .Xauthority
.bash_logout    Downloads        l2              .mozilla   .xsession-errors
.bashrc         examples.desktop l3              Music      .xsession-errors.old
.cache          f3               l3~             Pictures
```

Command: uname -p
Command description: Print the version number and copyright and quit.
syntax:uname -p
Output:

```
student@Lab-405-A-03:~$ uname -p
i686
```

Command: expr length
Command description: gives length of any string
syntax:expr length string
Output:

```
student@Lab-405-A-03:~$ expr length cspit
5
```

Command: rm -i filename
Command description: prompt before every removal . (before every file removal it will ask : do you want to remove or not?)
syntax: rm -i filename
Output:

```
student@Lab-405-A-03:~$ rm -i l2
rm: remove write-protected regular empty file ‘l2’? y
```

Command: cat -n
Command description: number to all output lines
syntax: cat -n filename
Output:

```
student@Lab-405-A-03:~$ cat -n l6
     1  hii
     2  how
     3  are
     4  you
```

Command: cat -E
Command description: put $ at last
syntax: cat -E
Output:

```
student@Lab-405-A-03:~$ cat -E l6
hii $
how$
are $
you$
```

Command: sort -r
Command description: reverse the result of comparisons
syntax: sort -r filename
Output:

```
student@Lab-405-A-03:~$ sort -r l6
you
how
hii
are
```

Command: ls -r

Command description: reverse order while sorting
syntax: ls-r filename
Output:

```
student@Lab-405-A-03:~$ ls -r l5
l5
```

Command: head -n filename | tail -m
Command description: display lines between n and m
syntax: head -A /path/to/file | tail -B
Output:

```
student@Lab-405-A-03:~$ head -2 l6 | tail -1
how
```

Command: char and esc
Command description: It will display all files starting with c.
syntax: any char and 4 times escape
Output:

```
student@Lab-405-A-03:~$ c
Display all 120 possibilities? (y or n)
c2ph                                col
c89                                 colcrt
c89-gcc                             colormgr
c99                                 colrm
c99-gcc                             column
cal                                 combinediff
calendar                            comm
calibrate_ppa                       command
```

Command: last -w
Command description: Display full user and domain anem in output.
syntax: last -w
Output:

```
student@Lab-405-A-03:~$ last -w
student  pts/0        :0               Tue Jan  1 01:16   still logged i
student  pts/0        :0               Tue Jan  1 00:47 - 01:16  (00:28)
student  pts/0        :0               Tue Jan  1 00:47 - 00:47  (00:00)
student  pts/0        :0               Tue Jan  1 00:06 - 00:47  (00:41)
student  :0           :0               Tue Jan  1 00:02   still logged i
```

Command: last -F
Command description: Login and Logout ime
syntax: last -F
Output:

```
student@Lab-405-A-03:~$ last -F
student  pts/0        :0               Tue Jan  1 01:16:32 2008   still
n
student  pts/0        :0               Tue Jan  1 00:47:55 2008 - Tue Ja
16:27 2008  (00:28)
student  pts/0        :0               Tue Jan  1 00:47:17 2008 - Tue Ja
47:23 2008  (00:00)
student  pts/0        :0               Tue Jan  1 00:06:13 2008 - Tue Ja
47:14 2008  (00:41)
```

Command: w -s
Command description: Use the short format. Don't print the login time, JCPU or PCPU times.
syntax: w- s
Output:

```
student@Lab-405-A-03:~$ w -s
 01:27:16 up  1:25,  2 users,  load average: 0.17, 0.23, 0.23
USER     TTY      FROM              IDLE WHAT
student  :0       :0               ?xdm?  init --user
student  pts/0    :0                4.00s w -s
```

Command: uptime -s
Command description: running time is in format. system up since, in yyyy-mm-dd MM:HH:SS format
syntax: uptime -s
Output:

```
student@Lab-405-A-03:~$ uptime -s
2008-01-01 00:01:37
```

Command: id -G
Command description: id of perticular user.
syntax: is -G
Output:

```
student@Lab-405-A-03:~$ id -G
1000 4 24 27 30 46 108 124
```

Command: history!n
Command description: Refer to command line n.
syntax: history!n
Output:

```
student@Lab-405-A-03:~$ history!n
historynano l4
historynano: command not found
```

Command: ps -u
Command description: Select by real user ID (RUID) or name. It selects the processes whose real user name or ID is in the userlist list.
syntax: ps -u
Output:

```
student@Lab-405-A-03:~$ ps -u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
student   4255  0.0  0.2   6924  4708 pts/0    Ss   01:16   0:00 bash
student   4345  0.0  0.0   4584   532 pts/0    T    01:17   0:00 cat -E
student   4800  0.0  0.1   5232  2416 pts/0    R+   01:31   0:00 ps -u
```

Command: ps -U root -u root u
Command description: To see every process running as root (real & effective ID) in user format:
syntax: ps -U root -u root u
Output:

```
student@Lab-405-A-03:~$  ps -U root -u root u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1   4460  3560 ?        Ss   00:01   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S    00:01   0:00 [kthreadd]
root         3  2.2  0.0      0     0 ?        S    00:01   2:03 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   00:01   0:00 [kworker/0:0H]
```

Command: ps -ejH
Command description: To print process tree.
syntax: ps -ejH
Output:

```
student@Lab-405-A-03:~$ ps -ejH
  PID  PGID   SID TTY          TIME CMD
    2     0     0 ?        00:00:00 kthreadd
    3     0     0 ?        00:02:04   ksoftirqd/0
    5     0     0 ?        00:00:00   kworker/0:0H
    7     0     0 ?        00:00:42   rcu_sched
```

Command: ps axZ
Command description: To get security info:
syntax: ps axZ
Output:

```
student@Lab-405-A-03:~$ ps axZ
LABEL                              PID TTY      STAT   TIME COMMAND
unconfined                           1 ?        Ss     0:01 /sbin/init
unconfined                           2 ?        S      0:00 [kthreadd]
unconfined                           3 ?        S      2:04 [ksoftirqd/0]
unconfined                           5 ?        S<     0:00 [kworker/0:0H]
unconfined                           7 ?        S      0:42 [rcu_sched]
```

Command: dmseg -K
Command description: Print kernel messages.
syntax: dmseg -K

Command: ps -d
Command description: Select all processes except session leaders.
syntax: ps -d
Output:

```
student@Lab-405-A-03:~$ ps -d
  PID TTY          TIME CMD
    2 ?        00:00:00 kthreadd
    3 ?        00:02:09 ksoftirqd/0
    5 ?        00:00:00 kworker/0:0H
    7 ?        00:00:43 rcu_sched
    8 ?        00:00:00 rcu_bh
    9 ?        00:00:00 migration/0
```

Command: w -o
Command description: Old style output.Prints blank space for idle times less than one minute.
syntax: w -o
Output:

```
student@Lab-405-A-03:~$ w -o
 01:38:51 up  1:37,  2 users,  load average: 0.03, 0.09, 0.16
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
student  :0       :0               00:02   ?xdm?  10:57m        init --user
student  pts/0    :0               01:16                         w -o
```

Command: id -u -r
Command description: display realtime user id.
syntax: id -u -r
Output:

```
student@Lab-405-A-03:~$ id -u -r
1000
```

Command: id -g -r
Command description: display realtime group id.
syntax: id -g -r
Output:

```
student@Lab-405-A-03:~$ id -g -r
1000
```

e. Study of Bash shell, Bourne shell and C shell in Linux operating system.

Bash shell : Bash is a Unix shell and command language written by Brian Fox for the GNU Project as a free software replacement for the Bourne shell. Bash is a command processor that typically runs in a text window, where the user types commands that cause actions. Bash can also read and execute commands from a file, called a script. Like all Unix shells, it supports filename globbing , piping, here documents, command substitution, variables, and control structures for condition-testing and iteration. The keywords, syntax and other basic features of the language are all copied from sh. Other features, e.g., history, are copied from csh and ksh. Bash is a POSIX-compliant shell, but with a number of extensions. The popularity of sh motivated programmers to develop a shell that was compatible with it, but with several enhancements. Linux systems still offer the sh shell, but "bash" -- the "Bourne-again Shell," based on sh -- has become the new default standard. One attractive feature of bash is its ability to run sh shell scripts unchanged. Shell scripts are complex sets of commands that automate programming and maintenance chores; being able to reuse these scripts saves programmers time. Conveniences not present with the original Bourne shell include command completion and a command history.

Bourne shell: Bourne shell (sh) is a shell, or command-line interpreter, for computer operating systems.The Bourne shell was the default shell for Version 7 Unix. Most Unix-like systems continue to have /bin/sh—which will be the Bourne shell, or a symbolic link or hard link to a compatible shell—even when other shells are used by most users. Bourne Shell is the oldest shell. It was written by Stephen Bourne at Bell Laboratories. The Bourne Shell has been the default shell for many UNIX like operating system and root users. It has been a de facto standard in the industry. The Bourne Shell has neither the interactive features, nor the complex programming constructs, of the C and Korn shells. Bourne shell is located at /bin/sh.A modified and advanced version may be installed in modern UNIX like operating systems. It may be a symbolic link more feature-rich shell than the Bourne shell.It is still used to write system initialization scripts located in /etc/rc.d/ or /usr/local/etc/rc.d or /etc/init.d/ scripts.

C shell : The C shell has three separate files which are used for customizing its environment. These three files are .cshrc, .login, and .logout. Because these files begin with a period (.) they do not usually appear when one types the **ls** command. In order to see all files beginning with periods, the **-a** option is used with the **ls** command.

f. **Study the basic environment variables: PATH, USER,HOME,SHELL**
Create your own environment variable.

Path : This command will display variables contain a colon separated list of directories in which the system looks for executable files.

Output:

```
student@Lab-405-A-10:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loc
al/games
```

User : This command will display which user will login.

Output:

```
student@Lab-405-A-10:~$ echo $USER
student
```

Home : This command will display the default path for the users home directory.

Output:

```
student@Lab-405-A-10:~$ echo $HOME
/home/student
```

Shell :This command will display the shell being used by the user.

Output:

```
student@Lab-405-A-10:~$ echo $SHELL
/bin/bash
```

Z : echo $ variable - To display value of a variable.

Env - Displays all environment variables.

Varibale name=variable value - create a new variable.

Unset – Remove a variable.

Export variable = value – To unset value of an environment variable.

Output:

```
student@Lab-405-A-10:~$ echo $z
10
```

```
student@Lab-405-A-10:~$ export z
student@Lab-405-A-10:~$ echo $z
10
student@Lab-405-A-10:~$ env
XDG_VTNR=7
```

```
z=10
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
```

```
student@Lab-405-A-10:~$ unset z
student@Lab-405-A-10:~$ echo $z

student@Lab-405-A-10:~$ env
XDG_VTNR=7
```

Practical – 2

Implement a Shell Scripts Program.

Program : Write a Shell Script which will display the "HELLO WORLD".

```
*a1.sh ×
echo  hello world
echo Operating system
```

Output:

```
student@Lab-405-A-10:~$ gedit a1.sh
student@Lab-405-A-10:~$ sh a1.sh
hello world
Operating system
```

Program : Write a Shell Script by using command line.

```
a4.sh ×
echo This is the command line print $1 $2 $3
```

Output:

```
student@Lab-405-A-10:~$ gedit a4.sh
student@Lab-405-A-10:~$ sh a4.sh hello operating system
This is the command line print hello operating system
```

Program : Write a Shell Script which calculates the arithmetic operation.

a2.sh

```
a=10;
b=20;
echo Addition is:
echo `expr $a + $b`
echo subtraction is:
echo `expr $a - $b`
echo Multiplication is:
echo `expr $a \* $b`
echo Division is:
echo `expr $a / $b`
echo Modulo is:
echo `expr $a % $b`
```

Output:

```
student@Lab-405-A-10:~$ sh a2.sh
Addition is:
30
subtraction is:
-10
Multiplication is:
200
Division is:
0
Modulo is:
10
```

Program : Write a Shell Script using if condition.

a3.sh (~) - gedit

Open Save Undo

a3.sh

```
a=10
b=10
if [ $a -eq $b ]
then
 echo $a and $b is equal
else
 echo $a and $b is unequal
fi
```

Output:

```
student@Lab-405-A-10:~$ gedit a3.sh
student@Lab-405-A-10:~$ sh a3.sh
10 and 10 is equal
```

Program : Write a Shell Script using nested if condition.

a5.sh

```
echo "Enter the number :"
read number
if [ $number = '100' ]
then
        echo "Enter a :"
        read a
        if [ $a = '10' ]
        then
        echo "valid number"
        else
        echo "invalid number"
        fi
else
echo not enter properly
fi
```

Output:

```
student@Lab-405-A-10:~$ gedit a5.sh
student@Lab-405-A-10:~$ sh a5.sh
Enter the number :
100
Enter a :
10
valid number
```

a. Write a script called hello which outputs the following:

- your username
- the time and date
- who is logged on
- also output a line of asterices (*********) after each section.

r7.sh

```
echo "**************************";
echo "Username :"`uname`
echo "**************************";
echo "Date :"`date`
echo "**************************";
echo "Logged In :"`who am i`
echo "**************************";
```

Output:

```
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ gedit r7.sh
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ sh r7.sh
*************************
Username :Linux
*************************
Date :Tue Aug 22 09:58:57 IST 2017
*************************
Logged In :student pts/2 2008-01-01 00:06 (:0)
*************************
```

b. Put the command hello into your .login file so that the script is executed every time that you log on.

```
student@Lab-405-A-03:~$ sudo gedit .bashrc
[sudo] password for student:
```

```
.bashrc ×
sh /home/student/k33.sh
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package
bash-doc)
```

Output:

```
student@Lab-405-A-03: ~
hello
student@Lab-405-A-03:~$
```

c. Write a shell program to simulate a simple calculator.

Output:

```
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ gedit r6.sh
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ sh r6.sh
5
7
+
12
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ sh r6.sh
2
3
-
-1
```

r6.sh

```
read a
read b
read c
case $c in
     +)
        echo `expr $a + $b`
        ;;
      -)
        echo `expr $a - $b`
        ;;
       \*)
        echo `expr $a \* $b`
        ;;
       \/)
        echo `expr $a \/ $b`
        ;;
       *)
esac
```

d. Write a script that will count the number of files in each of your Subdirectories.

Output:

r1.sh (~/Desktop) - gedit

Open Save Undo

r1.sh

```
c=0
for i in *
do
c=$(( c + 1 ))
done
echo $c
```

student@Lab-405-A-03: ~/Desktop

```
hello
student@Lab-405-A-03:~$ gedit r1.sh
student@Lab-405-A-03:~$ sh r1.sh
61
```

e. Write a shell script to combine any three text files into a single file (append them in the order as they appear in the arguments) and display the word count.

Output:

```
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ gedit r4.sh
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ sh r4.sh
enter first
i
enter second
j
enter third
h
f
f
Number of words
7 f
```

r4.sh (~/Pictures/15IT125/shell-script) - gedit

Open Save Undo

r4.sh

```
echo "enter first"
read a
echo "enter second"
read b
echo "enter third"
read c

read f

cat $a $b $c > $f

echo f

echo "Number of words"

wc -w $f
```

f. Write a shell program to count the following in a text file.

- Number of vowels in a given text file.
- Number of blank spaces.
- Number of characters.
- Number of symbols.
- Number of lines

Output:

```
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ gedit r8.sh
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ sh r8.sh
Enter the filename
i
Char
62 i
Lines
3 i
Speacial symbol
Vowel
i
a
e
i
u
```

r8.sh (~/Pictures/15IT125/shell-script) - gedit

Open Save Undo

r8.sh

```
echo Enter the filename
read fn
echo Char
wc -c $fn
echo Lines
wc -l $fn
echo Speacial symbol
grep -o -i "[@#$%^&*]" $fn
echo Vowel
grep -o -i "[aeiou]" $fn
```

g. Write a shell program to find the largest integer among the three integers given as arguments.

Output:

```
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ gedit r5.sh
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ sh r5.sh
enter first integer
5
enter second integer
1
enter third integer
9
9 is larger
```

r5.sh

```
echo "enter first integer"
read a
echo "enter second integer"
read b
echo "enter third integer"
read c

if [ $a -gt $b ]
then
        if [ $a -gt $c ]
        then
                echo $a" is larger"
        else
                echo $c" is larger"
        fi
else
        if [ $b -gt $c ]
        then
                echo $b" is larger"
        else
                echo $c" is larger"
        fi
fi
```

h. Write the shell program which produces a report from the output of ls – l in the following form:

- Only regular files, directories and symbolic links are printed.
- The file type and permissions are removed.
- A /character is appended to each directory name and the word DIR is printed at the beginning of the line.
- A @ character is appended to each symbolic link name and the word LINK is printed at the beginning of the line.
- At the end of the listing, the number of directories, symbolic links, regular files and the total size of regular files should be reported.

```
fc=0
dc=0
lc=0
size=0
for i in *
do
        if [ -f $i ]
then
        size=`ls -sh $i | awk '{print $1}'`
        echo "[FILE]" $i "  -->Size :" $size
```

```
            fi
done
for i in *
do
            if [ -d $i ]
then
            echo "[DIR]" $i "/"
            fi
done
for i in *
do
            if [ -L $i ]
then
            echo "[LINK]" $i "@"
fi
done
for i in *
do
            if [ -f $i ]
then
            fc=`expr $fc + 1`
            fi
            if [ -d $i ]
then
            dc=`expr $dc + 1`
            fi
            if [ -L $i ]
then
            lc=`expr $lc + 1`
            fi
done

echo "Total regular files : "$fc
echo "Total directories : "$dc
echo "Total links : "$lc
```

Output:

i. Write a shell script that searches for a single word pattern recursively in the current directory and displays the no. of times it occurred.

Output:

p1.sh

```
echo enter pattern
read a
%echo enter filename
%read b
echo result

for i in *
do
if [ -f $i ]
then
grep -r "$a" "$i" | wc -l
fi
done
```

p1.sh

```
echo enter pattern
read a
echo enter filename
read b
echo result
grep -r "$a" "$b" | wc -l
```

r3.sh

```
echo -n "enter filename::"
read fn
echo -n "enter one word pattern::"
read pattern
grep -c $pattern $fn
```

```
student@Lab-405-A-03:~$ gedit r3.sh
student@Lab-405-A-03:~$ sh r3.sh
enter filename::r31
enter one word pattern::hi
3
student@Lab-405-A-03:~$ cat r31
hi hello hi
hi
hi
```

j. Write a shell program to sort a given file which consists of a list of numbers, in ascending order.

Output:

```
file=""
echo -n "Enter Filename : "
read file

if [ ! -f $file ]
then
echo "$file not a file!"
exit 1
fi

sort -n $file
```

```
honey@honey-VirtualBox:~$ cat decen
3
2
1
```

```
honey@honey-VirtualBox:~$ sh a7.sh
Enter Filename : decen
1
2
3
```

k. A shell script, which is an interactive file – handling program with the following options: copy, remove, rename, link and exit. Once the user enters a choice, ask for the necessary information (like names of files, paths, etc.) and then carry out the necessary operation

Output:

r2.sh (~) - gedit

Open Save Undo

r2.sh

```
while [ 1 ]
do
echo "Enter choice :"
echo "1) copy "
echo "2) rename "
echo "3) remove "
echo "4) link "
echo "5) Exit "

read ch
echo "enter filename"
read fn
case $ch in
        1) echo "Enter new file to copy :"
        cp $fn $nfn ;;
        2) echo "enter new file name:"
        mv $fn $nfn ;;
        3) rm $fn ;;
        4) echo "enter filename to link :"
        ln -s $fn $fn2 ;;
        5) exit ;;
        *) echo "invalid"
esac
done
```

sh Tab Width: 8 Ln 12, Col 8 INS

```
student@Lab-405-A-03: ~
4) link
5) Exit
^C
student@Lab-405-A-03:~$ gedit r2.sh
student@Lab-405-A-03:~$ bash r2.sh
Enter choice :
1) copy
2) rename
3) remove
4) link
5) Exit
5
enter filename
f1
student@Lab-405-A-03:~$ 4
4: command not found
student@Lab-405-A-03:~$ bash r2.sh
Enter choice :
1) copy
2) rename
3) remove
4) link
5) Exit
4
```

l. Shell scripts that maintain a log file, consisting of log in and logout times of the user.

Output:
last -F | awk '{print $1"\t",$7"\t\t",$9}'

```
ubuntu@ubuntu:~$ last -F | awk '{print $1 "\t", $6 "\t\t",$8, $9, $10}'
ubuntu   20              2018 still logged
root     20              2018 - Mon
ubuntu   04:07:12                still logged in
ubuntu   04:07:12                still logged in
ubuntu   04:07:12                still logged in
ubuntu   04:07:12                still logged in
ubuntu   04:07:12                still logged in
ubuntu   04:07:12                still logged in
reboot   Aug             04:07:11 2018 -

wtmp     04:07:11
```

r10.sh (~/Pictures/15IT125/shell-script) - gedit

Open Save Undo

r10.sh

```
echo "log-file"

last -F | awk ' { print $1 " " $7 " " $9 }'
```

```
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ gedit r10.sh
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ sh r10.sh
log-file
student 09:54:38 still
student 09:54:21 -
student 09:54:21 -
student 00:42:53 -
student 00:06:10 still
```

m. Write a shell script that calculates shell script run time.

```
start_time=$(date +%s)

for i in {1..10}
do
echo $i
done

end_time=$(date +%s)

echo "Time duration: $((end_time - start_time)) secs."
```

Output:

```
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ gedit r11.sh
student@Lab-405-A-03:~/Pictures/15IT125/shell-script$ bash r11.sh
1
2
3
4
5
6
7
8
9
10
Time duration: 0 secs.
```

PRACTICAL - 3

Write a C program to list for every file in a directory, its inode number and file name.

Program:

fifth.c (~/Pictures/15IT125/c programs/fifth) - gedit

Open Save Undo

fifth.c

```c
#include<stdlib.h>
#include<stdio.h>
#include<string.h>

void main(int argc, char *argv[])
{
char d[50];
if(argc==2)
{
bzero(d,sizeof(d));
strcat(d,"ls ");
strcat(d,"-i ");
strcat(d,argv[1]);
system(d);
}
else
printf("\nInvalid No. of inputs");
}
```

C Tab Width: 8 Ln 17, Col 35 INS

Output:

```
student@Lab-405-A-03:~/Pictures/15IT125/c programs/fifth$ gedit fifth.c
student@Lab-405-A-03:~/Pictures/15IT125/c programs/fifth$ gcc -o fifth.out fifth
.c
student@Lab-405-A-03:~/Pictures/15IT125/c programs/fifth$ ./fifth.out f1
398035 f1
```

Practical- 4

Write a C program to create a child process and allow the parent to display "parent" and the child to display "child" on the screen.

Program:

sixth.c

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

int main()
{
int pid;
pid = fork();
if(pid<0)
{
printf("error");
exit(1);
}
else if(pid==0)
{
printf("hello i am child process\n");
printf("my pid is %d\n",getpid());
exit(0);
}
else
{
printf("hello i am parent process\n");
printf("my atcual pid is %d\n",getpid());
exit(0);
}
}
```

Output:

```
student@Lab-405-A-03:~/Pictures/15IT125/c programs$ gcc -o sixth.out sixth.c
student@Lab-405-A-03:~/Pictures/15IT125/c programs$ ./sixth.out
hello i am parent process
my atcual pid is 4780
hello i am child process
my pid is 4781
```

Practical – 5

Write a C program to implement grep system call.

Program:

nine.c (~/Pictures/15IT125/c programs) - gedit

```c
#include<stdio.h>
#include<string.h>
void main()
{
char fn[10],pat[10],temp[200];
FILE *fp;
printf("Enter file name\n");
scanf("%s",fn);
printf("Enter pattern to be searched\n");
scanf("%s",pat);
fp=fopen(fn,"r");
while(!feof(fp)) //eof = end of the file
{
fgets(temp,1000,fp);
if(strstr(temp,pat))
  printf("%s",temp);
}
fclose(fp);
}
```

C Tab Width: 8 Ln 4, Col 2 INS

Output:

```
student@Lab-405-A-03:~/Pictures/15IT125/c programs$ gedit nine.c
student@Lab-405-A-03:~/Pictures/15IT125/c programs$ gcc -o nine.out nine.c
student@Lab-405-A-03:~/Pictures/15IT125/c programs$ ./nine.out
Enter file name
k3
Enter pattern to be searched
ah
krunal shah
```

Practical – 6

Write a C program to implement inter process communication (IPC) using Semaphore.

Program:
Illustrate producer consumer problem.

pro-con.c (~) - gedit

Open Save Undo

pro-con.c

```c
#include<stdio.h>
#include<stdlib.h>

int mutex=1,full=0,empty=3,x=0;

int main()
{
    int n;
    void producer();
    void consumer();
    int wait(int);
    int signal(int);
    printf("\n1.Producer\n2.Consumer\n3.Exit");
    while(1)
    {
        printf("\nEnter your choice:");
        scanf("%d",&n);
        switch(n)
        {
            case 1:    if((mutex==1)&&(empty!=0))
                           producer();
                       else
                           printf("Buffer is full!!");
                       break;
            case 2:    if((mutex==1)&&(full!=0))
                           consumer();
                       else
                           printf("Buffer is empty!!");
                       break;
            case 3:
                       exit(0);
                       break;
        }
    }
```

C Tab Width: 8 Ln 20, Col 28 INS

pro-con.c (~) - gedit

Open Save Undo

pro-con.c

```c
        }
    }

    return 0;
}

int wait(int s)
{
    return (--s);
}

int signal(int s)
{
    return(++s);
}

void producer()
{
    mutex=wait(mutex);
    full=signal(full);
    empty=wait(empty);
    x++;
    printf("\nProducer produces the item %d",x);
    mutex=signal(mutex);
}

void consumer()
{
    mutex=wait(mutex);
    full=wait(full);
    empty=signal(empty);
    printf("\nConsumer consumes item %d",x);
    x--;
    mutex=signal(mutex);
}
```

C Tab Width: 8 Ln 50, Col 2 INS

Output:

```
student@Lab-405-A-03: ~
Welcome
student@Lab-405-A-03:~$ gedit pro-con.c
student@Lab-405-A-03:~$ gcc -o pro-con.out pro-con.c
student@Lab-405-A-03:~$ ./pro-con.out

1.Producer
2.Consumer
3.Exit
Enter your choice:1

Producer produces the item 1
Enter your choice:1

Producer produces the item 2
Enter your choice:1

Producer produces the item 3
Enter your choice:2

Consumer consumes item 3
Enter your choice:1

Producer produces the item 3
Enter your choice:1
Buffer is full!!
Enter your choice:2

Consumer consumes item 3
Enter your choice:2

Consumer consumes item 2
Enter your choice:2

Consumer consumes item 1
Enter your choice:2
Buffer is empty!!
Enter your choice:3
student@Lab-405-A-03:~$ gedit pro-con.c
```

Practical – 7

Deadlock:

a. Write a program for implementing Banker's algorithm.

p12_1.c (~) - gedit

Open Save Undo

p12_1.c

```c
#include<stdio.h>

void main()
{
int k=0,output[10],d=0,t=0,ins[5],i,avail[5],allocated[10][5],need[10]
[5],MAX[10][5],pno,P[10],j,rz, count=0;

printf("\n Enter the number of resources : ");
scanf("%d", &rz);

printf("\n enter the max instances of each resources\n");
for(i=0;i<rz;i++)
{  avail[i]=0;
printf("%c= ",(i+97));
scanf("%d",&ins[i]);
}
printf("\n Enter the number of processes : ");
scanf("%d", &pno);
printf("\n Enter the allocation matrix \n     ");

for(i=0;i<rz;i++)
printf(" %c",(i+97));
printf("\n");

for(i=0;i <pno;i++)
    {
            P[i]=i;
        printf("P[%d]  ",P[i]);

        for(j=0;j<rz;j++)
            {
                scanf("%d",&allocated[i][j]);
                avail[j]+=allocated[i][j];
```

C Tab Width: 8 Ln 81, Col 2 INS

```
        }
    }

    printf("\nEnter the MAX matrix \n     ");
    for(i=0;i<rz;i++)
    {
            printf(" %c",(i+97));
        avail[i]=ins[i]-avail[i];
    }
    printf("\n");
    for(i=0;i <pno;i++)
    {
        printf("P[%d]  ",i);
         for(j=0;j<rz;j++)
         scanf("%d", &MAX[i][j]);
    }

    printf("\n");
    A: d=-1;
    for(i=0;i <pno;i++)
    {
     count=0; t=P[i];
     for(j=0;j<rz;j++)
     {
     need[t][j] = MAX[t][j]-allocated[t][j];
     if(need[t][j]<=avail[j])
         count++;
     }
```

```
    if(count==rz)
    {
        output[k++]=P[i];
         for(j=0;j<rz;j++)
        avail[j]+=allocated[t][j];
```

```
    }
    else
     P[++d]=P[i];
    }

    if(d!=-1)
    {
     pno=d+1;
    goto A;
    }
    printf("\t <");
    for(i=0;i<k;i++)
    printf(" P[%d] ",output[i]);
    printf(">");

}
```

Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ gcc -o p12_1.out p12_1.c
student@Lab-405-A-03:~$ ./p12_1.out

 Enter the number of resources : 3

 enter the max instances of each resources
a= 10
b= 5
c= 7

 Enter the number of processes : 5

 Enter the allocation matrix
      a b c
P[0]  0 1 0
P[1]  2 0 0
P[2]  3 0 2
P[3]  2 1 1
P[4]  0 0 2

Enter the MAX matrix
      a b c
P[0]  7 5 3
P[1]  3 2 2
P[2]  9 0 2
P[3]  2 2 2
P[4]  4 3 3
< P[1]  P[3]  P[4]  P[0]  P[2]
```

b. Write a program that will surely go into the deadlock.

p12_2.c (~) - gedit

Open Save Undo

p12_2.c

```c
#include<stdio.h>
#include<pthread.h>
void *work1();
void *work2();
pthread_mutex_t lock1;
pthread_mutex_t lock2;

main()
{
pthread_t id1,id2;
pthread_mutex_init(&lock1,NULL);
pthread_mutex_init(&lock2,NULL);
pthread_create(&id1,NULL,work1,NULL);
pthread_create(&id2,NULL,work2,NULL);
pthread_exit(NULL);
}
void *work1()
{
printf("T1 before lock\n");
pthread_mutex_lock(&lock1);
sleep(2);
pthread_mutex_lock(&lock2);
pthread_mutex_unlock(&lock2);
pthread_mutex_unlock(&lock1);
}
void *work2()
{
printf("T2 before lock\n");
pthread_mutex_lock(&lock2);
sleep(2);
pthread_mutex_lock(&lock1);
pthread_mutex_unlock(&lock1);
pthread_mutex_unlock(&lock2);
```

C Tab Width: 8 Ln 1, Col 1 INS

```c
}
```

Output:

```
student@Lab-405-A-03:~$ gedit p12_2.c
student@Lab-405-A-03:~$ gcc -o p12_2.out p12_2.c -pthread
student@Lab-405-A-03:~$ ./p12_2.out
T1 before lock
T2 before lock
^C
```

Practical – 8

Write a program to solve dining philosophers' problem.

dining.c (~) - gedit

Open Save Undo

dining.c

```c
#include<stdio.h>
#include<semaphore.h>
#include<pthread.h>

#define N 5
#define THINKING 0
#define HUNGRY 1
#define EATING 2
#define LEFT (ph_num+4)%N
#define RIGHT (ph_num+1)%N

sem_t mutex;
sem_t S[N];

void * philospher(void *num);
void take_fork(int);
void put_fork(int);
void test(int);

int state[N];
int phil_num[N]={0,1,2,3,4};

int main()
{
    int i;
    pthread_t thread_id[N];
    sem_init(&mutex,0,1);
    for(i=0;i<N;i++)
        sem_init(&S[i],0,0);
    for(i=0;i<N;i++)
    {
        pthread_create(&thread_id[i],NULL,philospher,&phil_num[i]);
        printf("\nPhilosopher %d is thinkingn",i+1);
```

C Tab Width: 8 Ln 12, Col 13 INS

dining.c (~) - gedit

Open Save Undo

dining.c

```
    }
    for(i=0;i<N;i++)
        pthread_join(thread_id[i],NULL);
}

void *philospher(void *num)
{
    while(1)
    {
        int *i = num;
        sleep(1);
        take_fork(*i);
        sleep(0);
        put_fork(*i);
    }
}

void take_fork(int ph_num)
{
    sem_wait(&mutex);
    state[ph_num] = HUNGRY;
    printf("\nPhilosopher %d is Hungryn",ph_num+1);
    test(ph_num);
    sem_post(&mutex);
    sem_wait(&S[ph_num]);
    sleep(1);
}

void test(int ph_num)
{
    if (state[ph_num] == HUNGRY && state[LEFT] != EATING && state
[RIGHT] != EATING)
    {
```

C Tab Width: 8 Ln 48, Col 6 INS

dining.c (~) - gedit

Open Save Undo

dining.c

```
    state[ph_num] = HUNGRY;
    printf("\nPhilosopher %d is Hungryn",ph_num+1);
    test(ph_num);
    sem_post(&mutex);
    sem_wait(&S[ph_num]);
    sleep(1);
}

void test(int ph_num)
{
    if (state[ph_num] == HUNGRY && state[LEFT] != EATING && state
[RIGHT] != EATING)
    {
        state[ph_num] = EATING;
        sleep(2);
        printf("\nPhilosopher %d takes fork %d and %dn",ph_num+1,LEFT
+1,ph_num+1);
        printf("\nPhilosopher %d is Eatingn",ph_num+1);
        sem_post(&S[ph_num]);
    }
}

void put_fork(int ph_num)
{
    sem_wait(&mutex);
    state[ph_num] = THINKING;
    printf("\nPhilosopher %d putting fork %d and %d downn",ph_num+1,LEFT
+1,ph_num+1);
    printf("\nPhilosopher %d is thinkingn",ph_num+1);
    test(LEFT);
    test(RIGHT);
    sem_post(&mutex);
}
```

C Tab Width: 8 Ln 68, Col 83 INS

Output:

```
student@Lab-405-A-03: ~
student@Lab-405-A-03:~$ gedit dining.c
student@Lab-405-A-03:~$ gcc -o dining.out dining.c -pthread
student@Lab-405-A-03:~$ ./dining.out

Philosopher 1 is thinkingn
Philosopher 2 is thinkingn
Philosopher 3 is thinkingn
Philosopher 4 is thinkingn
Philosopher 5 is thinkingn
Philosopher 1 is Hungryn
Philosopher 1 takes fork 5 and 1n
Philosopher 1 is Eatingn
Philosopher 3 is Hungryn
Philosopher 3 takes fork 2 and 3n
Philosopher 3 is Eatingn
Philosopher 2 is Hungryn
Philosopher 5 is Hungryn
Philosopher 4 is Hungryn
Philosopher 1 putting fork 5 and 1 downn
Philosopher 1 is thinkingn
Philosopher 5 takes fork 4 and 5n
Philosopher 5 is Eatingn
Philosopher 3 putting fork 2 and 3 downn
Philosopher 3 is thinkingn
```