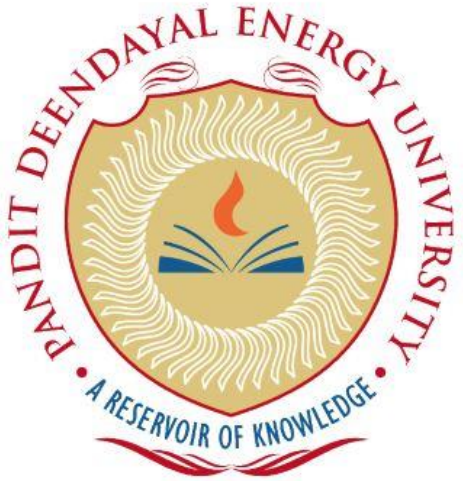
PANDIT DEENDAYAL ENERGY UNIVERSITY

SCHOOL OF TECHNOLOGY



Operating Systems Lab

20CP207P

LAB MANUAL

B.Tech. (Computer Science and Engineering)

Semester 4

Submitted To:

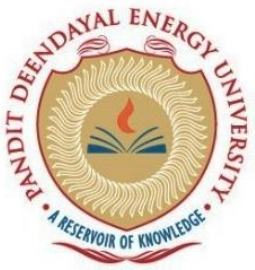
Dr. Hiren Thakkar

Submitted By:

HARSH SHAH

21BCP359

G11 Batch



SCHOOL OF TECHNOLOGY

PANDIT DEENDAYAL ENERGY UNIVERSITY

GANDHINAGAR, GUJARAT

CERTIFICATE

This is to certify that Mr. / Miss **Harsh Shah** Enrolment No. **21BCP359** of **4th Semester** degree course in **Computer Science & Engineering** has satisfactorily prepared and presenter his / her Term Work in **Operating Systems Lab (20CP207P)** within four walls of the laboratory of this Institute during **Jan 2023** to **May 2023.**

Date of Submission: 24-04-23

Dr. Hiren Kumar Thakkar

Assistant Professor

INDEX

Lab1: Basic Gate Operations

Execute the following Linux commands on Linux terminal and verify the output.

1. **whoami**

```
harsh@HP-Pavilion:~/ubuntu-new$ whoami
harsh
```

2. **pwd**

```
harsh@HP-Pavilion:~/ubuntu-new$ pwd
/home/harsh/ubuntu-new
```

3. **ls**

```
harsh@HP-Pavilion:~$ ls
ubuntu-new
```

4. **ls -r**

```
harsh@HP-Pavilion:~$ ls -R
.:
ubuntu-new
```

5. **ls -a**

```
harsh@HP-Pavilion:~$ ls -a
.  ..  .bash_logout  .bashrc  .motd_shown  .profile  ubuntu-new
```

6. **history**

```
harsh@HP-Pavilion:~$ history
    1  ls
    2  mkdir ubuntu-new
    3  ls
    4  cd ubuntu-new/
    5  clear
    6  whoami
    7  pwd
    8  ls
    9  mkdir
   10  ../
   11  cd ../
   12  ld
   13  ls
   14  ls -R
   15  ls -a
   16  history
```

7. **clear**

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/lab1$ clear
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/lab1$
```

8. **echo**

```
harsh@HP-Pavilion:~$ echo "This is Linux"
This is Linux
```

9. **touch**

```
harsh@HP-Pavilion:~$ touch file1.txt
harsh@HP-Pavilion:~$ ls
file1.txt  ubuntu-new
```

10. **rm**

```
harsh@HP-Pavilion:~$ rm file1.txt
harsh@HP-Pavilion:~$ ls
ubuntu-new
```

11. **mkdir**

```
harsh@HP-Pavilion:~/ubuntu-new$ mkdir
mkdir: missing operand
```

12. **rmdir**

```
harsh@HP-Pavilion:~$ rmdir ubuntu-new/
harsh@HP-Pavilion:~$ ls
```

13. **mv**

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/lab1$ mv file2.txt lab2
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/lab1$ ls
file1.txt  lab2
```

14. **cd**

```
harsh@HP-Pavilion:~$ cd lab1/
harsh@HP-Pavilion:~/lab1$ ls
file1.txt
```

15. **cmp**

```
harsh@HP-Pavilion:~/lab1$ cmp file1.txt file2.txt
file1.txt file2.txt differ: byte 13, line 1
```

16. cat

```
harsh@HP-Pavilion:~/lab1$ cat file1.txt
This is file1
```

17. cal

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ cal
    Januar 2023
So Mo Di Mi Do 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
```

18. cal -y

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ cal -y
                                2023
       Januar                     Februar                    März
So Mo Di Mi Do Fr Sa  So Mo Di Mi Do Fr Sa  So Mo Di Mi Do Fr Sa
 1  2  3  4  5  6  7               1  2  3  4            1  2  3  4
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   5  6  7  8  9 10 11
15 16 17 18 19 20 21  12 13 14 15 16 17 18  12 13 14 15 16 17 18
22 23 24 25 26 27 28  19 20 21 22 23 24 25  19 20 21 22 23 24 25
29 30 31              26 27 28              26 27 28 29 30 31


       April                      Mai                        Juni
So Mo Di Mi Do Fr Sa  So Mo Di Mi Do Fr Sa  So Mo Di Mi Do 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

        Juli                      August                   September
So Mo Di Mi Do Fr Sa  So Mo Di Mi Do Fr Sa  So Mo Di Mi Do 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

       Oktober                    November                   Dezember
So Mo Di Mi Do Fr Sa  So Mo Di Mi Do Fr Sa  So Mo Di Mi Do Fr Sa
 1  2  3  4  5  6  7               1  2  3  4                     1  2
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   3  4  5  6  7  8  9
15 16 17 18 19 20 21  12 13 14 15 16 17 18  10 11 12 13 14 15 16
22 23 24 25 26 27 28  19 20 21 22 23 24 25  17 18 19 20 21 22 23
29 30 31              26 27 28 29 30        24 25 26 27 28 29 30
                                            31
```

19. cal 2018

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ cal 2018
                                2018
       Januar                 Februar                  März
So Mo Di Mi Do Fr Sa  So Mo Di Mi Do Fr Sa  So Mo Di Mi Do Fr Sa
    1  2  3  4  5  6                  1  2  3                  1  2  3
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   4  5  6  7  8  9 10
14 15 16 17 18 19 20  11 12 13 14 15 16 17  11 12 13 14 15 16 17
21 22 23 24 25 26 27  18 19 20 21 22 23 24  18 19 20 21 22 23 24
28 29 30 31           25 26 27 28           25 26 27 28 29 30 31


       April                   Mai                    Juni
So Mo Di Mi Do Fr Sa  So Mo Di Mi Do Fr Sa  So Mo Di Mi Do Fr Sa
 1  2  3  4  5  6  7            1  2  3  4  5                  1  2
 8  9 10 11 12 13 14   6  7  8  9 10 11 12   3  4  5  6  7  8  9
15 16 17 18 19 20 21  13 14 15 16 17 18 19  10 11 12 13 14 15 16
22 23 24 25 26 27 28  20 21 22 23 24 25 26  17 18 19 20 21 22 23
29 30                 27 28 29 30 31        24 25 26 27 28 29 30


        Juli                  August                September
So Mo Di Mi Do Fr Sa  So Mo Di Mi Do Fr Sa  So Mo Di Mi Do Fr Sa
 1  2  3  4  5  6  7               1  2  3  4                     1
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   2  3  4  5  6  7  8
15 16 17 18 19 20 21  12 13 14 15 16 17 18   9 10 11 12 13 14 15
22 23 24 25 26 27 28  19 20 21 22 23 24 25  16 17 18 19 20 21 22
29 30 31              26 27 28 29 30 31     23 24 25 26 27 28 29
                                            30


      Oktober                November                Dezember
So Mo Di Mi Do Fr Sa  So Mo Di Mi Do Fr Sa  So Mo Di Mi Do Fr Sa
    1  2  3  4  5  6               1  2  3                     1
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   2  3  4  5  6  7  8
14 15 16 17 18 19 20  11 12 13 14 15 16 17   9 10 11 12 13 14 15
21 22 23 24 25 26 27  18 19 20 21 22 23 24  16 17 18 19 20 21 22
28 29 30 31           25 26 27 28 29 30     23 24 25 26 27 28 29
                                            30 31
```

20. passwd

```
harsh@HP-Pavilion:~/lab1$ passwd
Changing password for harsh.
Current password:
New password:
Retype new password:
passwd: password updated successfully
```

21. grep

```
harsh@HP-Pavilion:~/lab1$ grep "This is" file1.txt
This is file1
```

22. **free**

```
harsh@HP-Pavilion:~/lab1$ free
               total        used        free      shared  buff/cache   availa
ble
Mem:         3911656      333872     3345196        2512      232588     3364
352
Swap:        1048576           0     1048576
```

23. **uname**

```
harsh@HP-Pavilion:~/lab1$ uname
Linux
```

24. **uname -a**

```
harsh@HP-Pavilion:~/lab1$ uname -a
Linux HP-Pavilion 5.15.79.1-microsoft-standard-WSL2 #1 SMP Wed Nov 23 01:01:4
6 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
```

25. **uname -s**

```
harsh@HP-Pavilion:~/lab1$ uname -s
Linux
```

26. **uname -n**

```
harsh@HP-Pavilion:~/lab1$ uname -n
HP-Pavilion
```

27. **group**

```
harsh@HP-Pavilion:~/lab1$ groups
harsh adm dialout cdrom floppy sudo audio dip video plugdev netdev
```

28. **comm**

```
harsh@HP-Pavilion:~/lab1$ comm file1.txt file2.txt
This is file1
        This is file2
```

29. **date**

```
harsh@HP-Pavilion:~/lab1$ date
Mon Jan  9 13:17:50 IST 2023
```

30. date-d

```
harsh@HP-Pavilion:~/lab1$ date
Mon Jan  9 13:17:50 IST 2023
```

31. chmod

```
harsh@HP-Pavilion:~/lab1$ chmod +r file1.txt
```

32. wc

```
harsh@HP-Pavilion:~/lab1$ wc file1.txt
 1  3 14 file1.txt
```

Lab 2: Linux Commands - II

Execute the following Linux commands on Linux terminal and verify the output.

1. ifconfig

```
harsh@Ubuntu:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::1fbc:b8c8:401:5df9  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:79:13:85  txqueuelen 1000  (Ethernet)
        RX packets 298253  bytes 377118018 (377.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 70966  bytes 14051767 (14.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 16267  bytes 1819152 (1.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 16267  bytes 1819152 (1.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2. kill

```
harsh@Ubuntu:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

3. mount

```
harsh@Ubuntu:~$ mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=1973768k,nr_inodes=493442,mo
de=755,inode64)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmod
e=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,noexec,relatime,size=401824k,mode=755,
inode64)
/dev/sda3 on / type ext4 (rw,relatime,errors=remount-ro)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relat
ime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k,inode6
4)
```

```
harsh@Ubuntu:~$ mount -V
mount from util-linux 2.37.2 (libmount 2.37.2: selinux, smack, btrfs, verity, na
mespaces, assert, debug)
```

4. sort

```
harsh@Ubuntu:~/Desktop/g11$ cat file1.txt
c cat
z zebra
l lion
d dog
b buffalo
p parrot
harsh@Ubuntu:~/Desktop/g11$ sort file1.txt
b buffalo
c cat
d dog
l lion
p parrot
z zebra
```

5. export

```
harsh@Ubuntu:~/Desktop/g11$ export -p
declare -x COLORTERM="truecolor"
declare -x DBUS_SESSION_BUS_ADDRESS="unix:path=/run/user/1000/bus"
declare -x DESKTOP_SESSION="ubuntu"
declare -x DISPLAY=":0"
declare -x GDMSESSION="ubuntu"
declare -x GNOME_DESKTOP_SESSION_ID="this-is-deprecated"
declare -x GNOME_SETUP_DISPLAY=":1"
declare -x GNOME_SHELL_SESSION_MODE="ubuntu"
declare -x GNOME_TERMINAL_SCREEN="/org/gnome/Terminal/screen/23110a04_1016_43aa_9943_4e
7c129a61fd"
declare -x GNOME_TERMINAL_SERVICE=":1.99"
declare -x GTK_MODULES="gail:atk-bridge"
declare -x HOME="/home/harsh"
declare -x IM_CONFIG_PHASE="1"
declare -x LANG="en_IN"
declare -x LANGUAGE="en_IN:en"
declare -x LESSCLOSE="/usr/bin/lesspipe %s %s"
declare -x LESSOPEN="| /usr/bin/lesspipe %s"
declare -x LOGNAME="harsh"
```

6. ssh

```
harsh@Ubuntu:~/Desktop/g11$ ssh -l 192.168.1.1
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface]
           [-b bind_address] [-c cipher_spec] [-D [bind_address:]port]
           [-E log_file] [-e escape_char] [-F configfile] [-I pkcs11]
           [-i identity_file] [-J [user@]host[:port]] [-L address]
           [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
           [-Q query_option] [-R address] [-S ctl_path] [-W host:port]
           [-w local_tun[:remote_tun]] destination [command [argument ...]]
```

7. zip

```
harsh@Ubuntu:~/Desktop/g11$ ls
file1.txt  file2.txt
harsh@Ubuntu:~/Desktop/g11$ zip test.zip file1.txt file2.txt
  adding: file1.txt (stored 0%)
  adding: file2.txt (stored 0%)
harsh@Ubuntu:~/Desktop/g11$ ls
file1.txt  file2.txt  test.zip
```

8. unzip

```
harsh@Ubuntu:~/Desktop/g11$ unzip test.zip
Archive:  test.zip
replace file1.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
 extracting: file1.txt
 extracting: file2.txt
harsh@Ubuntu:~/Desktop/g11$ ls
file1.txt  file2.txt  test.zip
```

9. ps

```
harsh@Ubuntu:~/Desktop/g11$ ps
    PID TTY          TIME CMD
   3413 pts/0    00:00:00 bash
   3444 pts/0    00:00:00 ps
```

10. uname

```
harsh@Ubuntu:~/Desktop/g11$ uname
Linux
```

11. chown

```
harsh@Ubuntu:~/Desktop/g11$ chown harsh file1.txt
harsh@Ubuntu:~/Desktop/g11$ ls -l file1.txt
-rw-rw-r-- 1 harsh harsh 46 Jan 16 20:53 file1.txt
```

12. wget

```
harsh@Ubuntu:~/Desktop/g11$ wget https://sot.pdpu.ac.in/downloads/UG_Curriculum_
CSED_2020-24.pdf
--2023-01-16 22:36:38--  https://sot.pdpu.ac.in/downloads/UG_Curriculum_CSED_202
0-24.pdf
Resolving sot.pdpu.ac.in (sot.pdpu.ac.in)... 203.77.200.33
Connecting to sot.pdpu.ac.in (sot.pdpu.ac.in)|203.77.200.33|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3799836 (3.6M) [application/pdf]
Saving to: 'UG_Curriculum_CSED_2020-24.pdf'

UG_Curriculum_CSED_20 100%[=======================>]   3.62M  1.19MB/s    in 3.
0s

2023-01-16 22:36:42 (1.19 MB/s) - 'UG_Curriculum_CSED_2020-24.pdf' saved [379983
6/3799836]
```

13. ufw

```
root@Ubuntu:/home/harsh# sudo ufw status verbose
Status: inactive
root@Ubuntu:/home/harsh# sudo ufw enable
Firewall is active and enabled on system startup
root@Ubuntu:/home/harsh# sudo ufw disable
Firewall stopped and disabled on system startup
```

14. traceroute

```
root@Ubuntu:/home/harsh# traceroute www.pdpu.ac.in
traceroute to www.pdpu.ac.in (203.77.200.33), 30 hops max, 60 byte packets
 1  _gateway (10.0.2.2)  3.291 ms  3.222 ms  3.140 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  *^C
```

15. service

```
harsh@Ubuntu:~/Desktop$ service --status-all
 [ + ]  acpid
 [ - ]  alsa-utils
 [ - ]  anacron
 [ + ]  apparmor
 [ + ]  apport
 [ + ]  avahi-daemon
 [ - ]  bluetooth
 [ - ]  console-setup.sh
 [ + ]  cron
 [ + ]  cups
 [ + ]  cups-browsed
 [ + ]  dbus
 [ + ]  gdm3
 [ - ]  grub-common
 [ - ]  hwclock.sh
 [ + ]  irqbalance
 [ + ]  kerneloops
 [ - ]  keyboard-setup.sh
 [ + ]  kmod
 [ - ]  open-vm-tools
 [ + ]  openvpn
 [ - ]  plymouth
 [ + ]  plymouth-log
 [ + ]  procps
 [ - ]  pulseaudio-enable-autospawn
 [ - ]  rsync
 [ - ]  saned
 [ - ]  speech-dispatcher
 [ - ]  spice-vdagent
 [ + ]  udev
 [ + ]  ufw
 [ + ]  unattended-upgrades
 [ - ]  uuidd
 [ - ]  whoopsie
 [ - ]  x11-common
```

16. alias

```
harsh@Ubuntu:~/Desktop$ alias c="clear"
harsh@Ubuntu:~/Desktop$ alias
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo
 error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'
')"'
alias c='clear'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
```

17. dd

```
harsh@Ubuntu:~/Desktop/g11$ touch abc.txt xyz.txt
harsh@Ubuntu:~/Desktop/g11$ ls
abc.txt  file2.txt  test.zip  UG_Curriculum_CSED_2020-24.pdf  xyz.txt
harsh@Ubuntu:~/Desktop/g11$ echo "This is abc.txt" >> abc.txt
harsh@Ubuntu:~/Desktop/g11$ echo "This is xyz.txt" >> xyz.txt
harsh@Ubuntu:~/Desktop/g11$ cat abc.txt xyz.txt
This is abc.txt
This is xyz.txt
harsh@Ubuntu:~/Desktop/g11$ dd if=abc.txt of=xyz.txt
0+1 records in
0+1 records out
16 bytes copied, 0.000778132 s, 20.6 kB/s
harsh@Ubuntu:~/Desktop/g11$ cat xyz.txt
This is abc.txt
```

18. whereis

```
harsh@Ubuntu:~/Desktop$ whereis bash
bash: /usr/bin/bash /usr/share/man/man1/bash.1.gz
```

19. whatis

```
harsh@Ubuntu:~/Desktop$ whatis cp rm touch
cp (1)               - copy files and directories
rm (1)               - remove files or directories
touch (1)            - change file timestamps
```

20. diff

```
harsh@Ubuntu:~/Desktop/g11$ diff file1.txt file2.txt
1,6d0
< c cat
< z zebra
< l lion
< d dog
< b buffalo
< p parrot
```

21. ln

```
harsh@Ubuntu:~/Desktop/g11$ ln -s file1.txt link.txt
harsh@Ubuntu:~/Desktop/g11$ ls -l link.txt
lrwxrwxrwx 1 harsh harsh 9 Jan 17 19:26 link.txt -> file1.txt
```

22. top

```
harsh@Ubuntu:~/Desktop/g11$ top

top - 19:27:39 up 24 min,  1 user,  load average: 0.00, 0.00, 0.03
Tasks: 185 total,   1 running, 184 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.7 us,  0.7 sy,  0.0 ni, 98.2 id,  0.2 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :   3924.0 total,   2477.1 free,    671.5 used,    775.5 buff/cache
MiB Swap:   2680.0 total,   2680.0 free,      0.0 used.   3007.5 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1446 harsh     20   0 3954140 309284 125224 S   3.0   7.7   0:55.39 gnome-s+
    280 root     -51   0       0      0      0 S   0.3   0.0   0:01.09 irq/18-+
   1377 harsh     39  19  642080  28164  18776 S   0.3   0.7   0:01.26 tracker+
   1957 harsh     20   0  581280  61368  41844 S   0.3   1.5   0:11.63 gnome-t+
      1 root      20   0  166712  11860   8284 S   0.0   0.3   0:01.93 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.02 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par+
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 slub_fl+
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
      8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker+
      9 root      20   0       0      0      0 I   0.0   0.0   0:03.15 kworker+
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_perc+
```

23. useradd

```
root@Ubuntu:/home/harsh/Desktop/g11# useradd harsh2
Password:
```

24. man

```
root@Ubuntu:/home/harsh/Desktop/g11# man -f sleep
sleep (1)            - delay for a specified amount of time
sleep (3)            - sleep for a specified number of seconds
```

25. cp

```
harsh@Ubuntu:~/Desktop/g11$ cat file2.txt
This is file2.txt
harsh@Ubuntu:~/Desktop/g11$ cp file1.txt file2.txt
harsh@Ubuntu:~/Desktop/g11$ cat file2.txt
c cat
z zebra
l lion
d dog
b buffalo
p parrot
```

26. ln

```
harsh@Ubuntu:~/Desktop/g11$ ln -s file1.txt link.txt
harsh@Ubuntu:~/Desktop/g11$ ls -l link.txt
lrwxrwxrwx 1 harsh harsh 9 Jan 17 19:26 link.txt -> file1.txt
```

27. netstat

```
harsh@Ubuntu:~/Desktop/g11$ netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG        0 0          0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 enp0s3
```

28. nslookup

```
harsh@Ubuntu:~/Desktop/g11$ nslookup yahoo.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   yahoo.com
Address: 98.137.11.163
Name:   yahoo.com
Address: 74.6.143.26
Name:   yahoo.com
Address: 74.6.143.25
Name:   yahoo.com
Address: 74.6.231.20
Name:   yahoo.com
Address: 74.6.231.21
Name:   yahoo.com
Address: 98.137.11.164
Name:   yahoo.com
Address: 2001:4998:44:3507::8001
Name:   yahoo.com
Address: 2001:4998:124:1507::f001
Name:   yahoo.com
Address: 2001:4998:24:120d::1:0
Name:   yahoo.com
Address: 2001:4998:44:3507::8000
Name:   yahoo.com
Address: 2001:4998:24:120d::1:1
Name:   yahoo.com
```

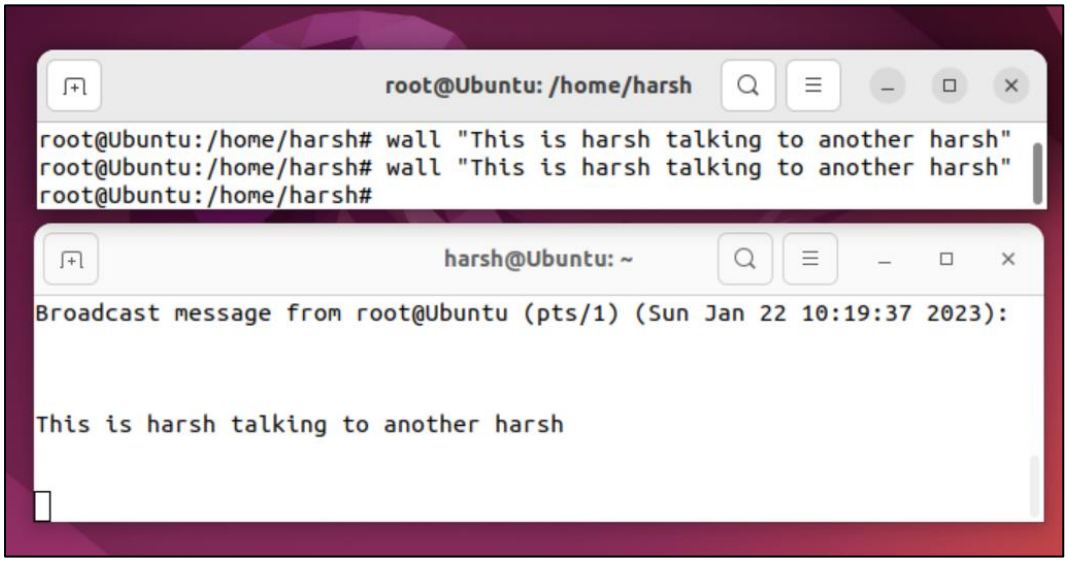
29. dig

```
harsh@Ubuntu:~$ dig google.com

; <<>> DiG 9.18.1-1ubuntu1.1-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33320
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             0       IN      A       142.251.42.14

;; Query time: 12 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Thu Jan 19 10:52:59 IST 2023
;; MSG SIZE  rcvd: 55
```

30. uptime

```
harsh@Ubuntu:~$ uptime
 10:54:49 up 12 min,  1 user,  load average: 2.10, 1.38, 0.80
```

31. wall



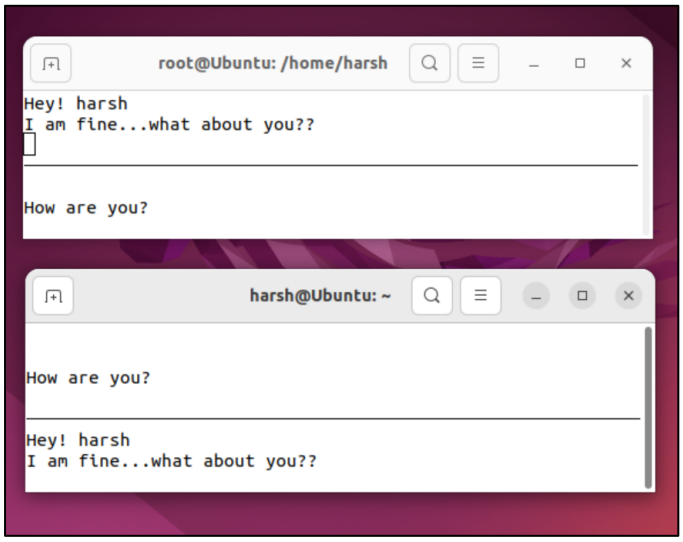
32. mesg

```
harsh@Ubuntu:~/Desktop/g11$ mesg
is y
harsh@Ubuntu:~/Desktop/g11$ mesg n
harsh@Ubuntu:~/Desktop/g11$ mesg
is n
```

33. write

```
root@Ubuntu:/home/harsh# write harsh
Hey Harsh! Break in 30 mins
Regards, HSroot@Ubuntu:/home/harsh# write harsh
Hey Harsh! Break in 30 mins
Regards, HS
root@Ubuntu:/home/harsh# w
 11:26:10 up 44 min,  3 users,  load average: 0.59, 0.63, 0.83
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
harsh    tty2     tty2             10:43   44:36   0.07s  0.07s /usr/libexec/gn
harsh    pts/1    -                11:18    1.00s  0.18s  0.07s sudo login hars
harsh    pts/1    -                11:19    1.00s  0.18s  0.06s -bash
```

34. talk



35. w

```
root@Ubuntu:/home/harsh# w
 11:33:07 up 51 min,  3 users,  load average: 3.95, 1.91, 1.25
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
harsh    tty2     tty2             10:43   51:33   0.07s  0.07s /usr/libexec/gn
harsh    pts/1    -                11:18    1.00s  0.41s  0.11s sudo login hars
harsh    pts/1    -                11:19    1.00s  0.41s  0.06s -bash
```

36. rename

```
harsh@Ubuntu:~/Desktop/g11$ cat file1.txt
c cat
z zebra
l lion
d dog
b buffalo
p parrot
harsh@Ubuntu:~/Desktop/g11$ cat file2.txt
This is file2.txt

harsh@Ubuntu:~/Desktop/g11$ mv file1.txt file2.txt
harsh@Ubuntu:~/Desktop/g11$ cat file1.txt
cat: file1.txt: No such file or directory
harsh@Ubuntu:~/Desktop/g11$ cat file2.txt
c cat
z zebra
l lion
d dog
b buffalo
p parrot
```

37. free

```
harsh@Ubuntu:~/Desktop/g11$ free
               total        used        free      shared  buff/cache   available
Mem:         4018208     1413360      683316       55136     1921532     2318144
Swap:        2744316           0     2744316
```

Lab 3: Linux Commands - II

Write the following Scripts and Execute in Linux Terminal

1. **Write a shell script to print your name.**

```
read -p "Enter your Name: " PERSON
echo "Hello! $PERSON"
```

```
harsh@Ubuntu:~/Desktop/lab3$ sh hello.sh
Enter your Name: Harsh
Hello! Harsh
```

2. **Write a shell script to find whether a number is even or odd.**

```
read -p "Hello user, Enter a number to check whether it is odd or even: " num
if [ "$(expr $num % 2)" -eq 0 ]
then
        echo "$num is even"
else
        echo "$num is odd"
fi
```

```
harsh@Ubuntu:~/Desktop/lab3$ sh evenodd.sh
Hello user, Enter a number to check whether it is odd or even: 55
55 is odd
harsh@Ubuntu:~/Desktop/lab3$ sh evenodd.sh
Hello user, Enter a number to check whether it is odd or even: 44
44 is even
```

3. **Write a script to print a table of a given number.**

```
echo "--- Enter Number to Generate Multiplication Table ---"
read -p "Enter the number : " num
i=1
while [ $i -le 10 ]
do
echo " $num * $i = `expr $num \* $i ` "
i=`expr $i + 1`
done
```

```
harsh@Ubuntu:~/Desktop/lab3$ sh table.sh
--- Enter Number to Generate Multiplication Table ---
Enter the number : 13
 13 * 1 = 13
 13 * 2 = 26
 13 * 3 = 39
 13 * 4 = 52
 13 * 5 = 65
 13 * 6 = 78
 13 * 7 = 91
 13 * 8 = 104
 13 * 9 = 117
 13 * 10 = 130
```

4. **Write a shell script to check whether a given no. is prime or not.**

```
read -p "Enter a number: " num
i=2
f=0
while [ $i -le `expr $num / 2` ]
```

```
do
if [ `expr $num % $i` -eq 0 ]
then
f=1
fi
i=`expr $i + 1`
done
if [ $f -eq 1 ]
then
echo "$num is composite (not prime)"
else
echo "$num is Prime"
fi
```

```
harsh@Ubuntu:~/Desktop/lab3$ sh prime.sh
Enter a number: 12
12 is composite (not prime)
harsh@Ubuntu:~/Desktop/lab3$ sh prime.sh
Enter a number: 13
13 is Prime
```

5. **Write a shell script to find the simple interest.**

```
read -p "Enter the principle value: " p
read -p "Enter the rate of interest: " r
read -p "Enter the time period: " t
i=` expr $p \* $t \* $r `
j=` expr $i / 100 `
echo "Simple Interest is: $j"
```

```
harsh@Ubuntu:~/Desktop/lab3$ sh simpleinterest.sh
Enter the principle value: 20000
Enter the rate of interest: 5
Enter the time period: 2
Simple Interest is: 2000
```

6. **Write a shell script to find sum of 'n' numbers.**

```
read -p "How many numbers do you want to sum up : " N
i=1
sum=0
while [ $i -le $N ]
do
  read -p "Enter number $i : " num
  sum=$((sum + num))
  i=$((i + 1))
done
echo "Sum :" $sum
```

```
harsh@Ubuntu:~/Desktop/lab3$ sh nsum.sh
How many numbers do you want to sum up : 6
Enter number 1 : 8
Enter number 2 : 6
Enter number 3 : 9
Enter number 4 : 4
Enter number 5 : 1
Enter number 6 : 3
Sum : 31
```

7. **Write a shell script to find the largest number of three numbers.**

```
read -p "Enter Num1 : " num1
read -p "Enter Num2 : " num2
read -p "Enter Num3 : " num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
then
   echo "The largest numbers among $num1, $num2, $num3 is : "$num1
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
then
   echo "The largest numbers among $num1, $num2, $num3 is : "$num2
else
   echo "The largest numbers among $num1, $num2, $num3 is : "$num3
fi
```

```
harsh@Ubuntu:~/Desktop/lab3$ sh largest3.sh
Enter Num1 : 6
Enter Num2 : 8
Enter Num3 : 4
The largest numbers among 6, 8, 4 is : 8
```

Lab 4: Linux Basic Scripting - 2

1. **Write a menu driven shell script, which will print the following menu and execute the given task.**
   - **Display a calendar of current month**
   - **Display today's date and time**
   - **Display username those are currently logged in the system**
   - **Display your name at the given x,y position.**
   - **Display your terminal number.**

```
echo "";
i=0
while [ $i != 6 ]
do
    echo "1. Display calender of current Month"
    echo "2. Display current date and time"
    echo "3. Display usernames of those who are currently logged in the system"
    echo "4. Display your name at given x, y position"
    echo "5. Display Terminal Number"
    echo "6. Exit"
    read -p "Choose your option & enter corresponding value: " ch
    echo "";

    case "$ch" in
        1) cal;;
        2) date;;
        3) whoami;;
        4) row=$(tput lines)
           col=$(tput cols)
           echo "Terminal Window has Rows: $row Cols: $col"

           read -p "X position: " x
           read -p "Y position: " y
           read -p "Enter name: " name
           tput cup $x $y
           echo "$name";;
        5) tty;;
        6) exit;;
        *) echo "Enter valid choice";;
    esac
    echo "";
done  SS
```

Output

```
1. Display calender of current Month
2. Display current date and time
3. Display usernames of those who are currently logged in the system
4. Display your name at given x, y position
5. Display Terminal Number
6. Exit
Choose your option & enter corresponding value: 1

   February 2023
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
```

```
1. Display calender of current Month
2. Display current date and time
3. Display usernames of those who are currently logged in the system
4. Display your name at given x, y position
5. Display Terminal Number
6. Exit
Choose your option & enter corresponding value: 2

Friday 03 February 2023 06:11:08 PM IST

1. Display calender of current Month
2. Display current date and time
3. Display usernames of those who are currently logged in the system
4. Display your name at given x, y position
5. Display Terminal Number
6. Exit
Choose your option & enter corresponding value: 3

harsh
```

```
1. Display calender of current Month
2. Display current date and time
3. Display usernames of those who are currently logged in the system
4. Display your name at given x, y position
5. Display Terminal Number
6. Exit
Choose your option & enter corresponding value: 4

Terminal Window has Rows: 37 Cols: 166
X position: 35
Y position: 100
Enter name: Ubuntu                                                    Ubuntu
```

```
1. Display calender of current Month
2. Display current date and time
3. Display usernames of those who are currently logged in the system
4. Display your name at given x, y position
5. Display Terminal Number
6. Exit
Choose your option & enter corresponding value: 5

/dev/pts/0
```

2. Write a shell script which will generate first n Fibonacci numbers.

```
echo "";
echo "Program to print Fibonacci Sequence of n numbers"
fib1=1
fib2=1
fib3=1

read -p "Enter the number of terms : " n
echo -n "Fibonacci Sequence : 1 1 "

for ((i=2 ; i<n ; i++))
do
    fib1=`expr $fib2`
    fib2=`expr $fib3`
    fib3=`expr $fib1 + $fib2`
    echo -n "$fib3 "
done
echo "";
echo "";
```

Output:

```
harsh@Ubuntu:~/Desktop/OS Lab Course/lab4$ bash fibonacci.sh

Program to print Fibonacci Sequence of n numbers
Enter the number of terms : 8
Fibonacci Sequence : 1 1 2 3 5 8 13 21
```

3. **Shell Script to print half pyramids using numbers.**

```
#!/bin/bash

read -p "Enter height of pyramid: " rows
number=1

for((i=1; i<=rows; i++))
do
  for((j=1; j<=i; j++))
  do
    echo -n "$number "
    number=$((number + 1))
  done
  number=1
  echo
done
```

Output:

```
harsh@Ubuntu:~/Desktop/OS Lab Course/lab4$ bash halfPyramid.sh
Enter height of pyramid: 6
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
```

4. **Write a shell script to find the reverse of a given number.**

```
#!/bin/bash

read -p "Enter the number: " n
c=0
while [ $n -gt 0 ]
do
a=`expr $n % 10 `
n=`expr $n / 10 `
c=`expr $c \* 10 + $a`
done
echo "Reversed number: $c"
```

Output:

```
harsh@Ubuntu:~/Desktop/OS Lab Course/lab4$ bash reverse.sh
Enter the number: 12345
Reversed number: 54321
```

5. **Write a shell script to find the sum of two floating point numbers.**

```
echo "";
echo "Program to add 2 Float nos"

read -p "Enter number 1: " a
read -p "Enter number 2: " b

echo -n "Sum : $a + $b = "
echo "$a + $b" | bc
```

Output:

```
harsh@Ubuntu:~/Desktop/OS Lab Course/lab4$ bash floatAdd.sh

Program to add 2 Float nos
Enter number 1: 2.5
Enter number 2: 6.4
Sum : 2.5 + 6.4 = 8.9
```

6. **Write a shell script to make the following operations menu based:**
   - **Addition**
   - **Subtraction**
   - **Multiplication**
   - **Division**

```
echo ""
i=0
ch=0

read -p "Enter number 1: " a
read -p "Enter number 2: " b

while [ $ch != 5 ]
do
    echo "1. Addition"
    echo "2. Subtraction"
    echo "3. Multiplication"
    echo "4. Division"
    echo "5. Exit"
    read -p "Choose your option & enter corresponding value: " ch
    echo "";

    case "$ch" in
        1) echo "Addition: `expr $a + $b`";;
        2) echo "Subtraction: `expr $a - $b`";;
        3) echo "Multiplication: `expr $a \* $b`";;
        4) echo "Division: `expr $a / $b`";;
        5) exit;;
        *) echo "PLEASE Enter valid choice";;
    esac
    echo "";

done
```

Output:

```
harsh@Ubuntu:~/Desktop/OS Lab Course/lab4$ bash menuOperations.sh

Enter number 1: 65
Enter number 2: 13
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Choose your option & enter corresponding value: 1

Addition: 78

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Choose your option & enter corresponding value: 2

Subtraction: 52
```

```
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Choose your option & enter corresponding value: 3

Multiplication: 845

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Choose your option & enter corresponding value: 4

Division: 5

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Choose your option & enter corresponding value: 5
```

7. **Write a shell script to find the sum of all digits for a given number.**

```
echo -n "Enter a number: "
read num

temp=$num
sum=0
while ((num > 0))
do
    sum=$((num%10+sum))
    num=$((num/10))
done
echo "Sum of digits of $temp is $sum"
```

Output:

```
harsh@Ubuntu:~/Desktop/OS Lab Course/lab4$ bash sumOfDigit.sh
Enter a number: 12345
Sum of digits of 12345 is 15
```

8. **Write a shell script to find the factorial of a given number.**

```
echo -n "Enter a number: "
read num

fact=1
for ((i = num; i > 0; i --))
do
    fact=$((fact*i))
done
echo "Factorial of $num: $fact"
```

Output:

```
harsh@Ubuntu:~/Desktop/OS Lab Course/lab4$ bash factorial.sh
Enter a number: 6
Factorial of 6: 720
```

9. **Write a shell script which prints "invalid no. of arguments" if more than 5 command line arguments otherwise print "valid no. of arguments".**

```
#!/bin/bash

if [ $# -gt 5 ]; then
  echo "Invalid no. of arguments"
else
  echo "Valid no. of arguments"
fi
```

Output:

```
harsh@Ubuntu:~/Desktop/OS Lab Course/lab4$ bash arguement.sh arg1 arg2 arg3 arg4 arg5
Valid no. of arguments
harsh@Ubuntu:~/Desktop/OS Lab Course/lab4$ bash arguement.sh arg1 arg2 arg3 arg4 arg5 arg6
Invalid no. of arguments
```

10. **Write a shell script that changes text to uppercase.**

```
echo ""

echo -n "Enter some text: "
read text

echo "$text" | tr '[:lower:]' '[:upper:]'
```

Output:

```
harsh@Ubuntu:~/Desktop/OS Lab Course/lab4$ bash uppercase.sh

Enter some text: hello this is ubuntu.
HELLO THIS IS UBUNTU.
```

Lab 5: Fork System Call

A system call is a way for a user program to interface with the operating system. The program requests several services, and the OS responds by invoking a series of system calls to satisfy the request. A system call can be written in assembly language or a high-level language like C or Pascal. System calls are predefined functions that the operating system may directly invoke if a high-level language is used.

The Application Program Interface (API) connects the operating system's functions to user programs. It acts as a link between the operating system and a process, allowing user-level programs to request operating system services. The kernel system can only be accessed using system calls. System calls are required for any programs that use resources. One of the most frequently used system call is the fork system call. The fork system call is used to create a new process. The newly created process is the child process. The process which calls fork and creates a new process is the parent process. The child and parent processes are executed concurrently.

But the child and parent processes reside on different memory spaces. These memory spaces have same content and whatever operation is performed by one process will not affect the other process. Once the child processes are created, now both the processes will have the same Program Counter (PC), so both processes will point to the same next instruction. The files opened by the parent process will be the same for child process.

Difference between Process Id's of Child & Parent process

1. The process ID of the child process is a unique process ID which is different from the IDs of all other existing processes.
2. The Parent process ID will be the same as that of the process ID of child's parent.

Although the child process is created from the parent process, they differ from each other in several ways. Some of the key properties of the parent and child processes are listed below:

Properties of child Process:

1. The CPU counters and the resource utilizations are initialized to reset to zero.
2. When the parent process is terminated, child processes do not receive any signal.
3. The thread used to call fork() creates the child process. So, the address of the child process will be the same as that of parent.
4. The file descriptor of parent process is inherited by the child process. For example, the offset of the file or status of flags and the I/O attributes will be shared among the file descriptors of child and parent processes. So, file descriptor of parent class will refer to same file descriptor of child class.
5. The open message queue descriptors of parent process are inherited by the child process. For example, if a file descriptor contains a message in parent process the same message will be present in the corresponding file descriptor of child process. So, we can say that the flag values of these file descriptors are same.
6. Similarly open directory streams will be inherited by the child processes.
7. The default Timer slack value of the child class is same as the current timer slack value of parent class.

Properties which are not inherited by child process:

1. Memory locks
2. The pending signal of a child class is empty.
3. Process associated record locks.
4. Asynchronous I/O operations and I/O contents.
5. Directory change notifications.
6. Timers such as alarm(), setTimer() are not inherited by the child class

Fork() call:

The fork() system call is used to create a new process by duplicating the calling process. The fork() system call is made by the parent process, and if it is successful, a child process is created. It does not accept any parameters and it returns an integer value. After the creation of a new child process, both processes then execute the next command following the fork system call. Therefore, it is a must to separate the parent process from the child by checking the returned value of the fork():

- **Negative:** A child process could not be successfully created if the fork() returns a negative value.
- **Zero:** A new child process is successfully created if the fork() returns a zero
- **Positive:** The positive value is the process ID of a child's process to the parent. The process ID is the type of **pid_t**.

Syntax:

pid_t fork(void);

Implementation

However, it can also be used in programming languages such as C. Consider the following basic implementation to understand the syntax and working of fork() call.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int main()
{
        fork();
        printf("Hello World \n");
        return 0;
}
```

```
harsh@Ubuntu:~/Desktop/OS Lab Course/lab5$ gcc fork.c -o fork1
harsh@Ubuntu:~/Desktop/OS Lab Course/lab5$ ./fork1
Hello World
Hello World
```

In the above program, the fork() function is used, which will create a new child process. Once the child process has been created, the child process and the parent process will both point to the next command. In this manner, the remaining commands will be executed 2^n times, where n represents the number of fork() system calls. Consider another implementation below:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int main()
{
        fork();
        fork();
        fork();
        printf("Hello World \n");
        return 0;
}
```

```
harsh@Ubuntu:~/Desktop/OS Lab Course/lab5$ gcc fork.c -o fork1
harsh@Ubuntu:~/Desktop/OS Lab Course/lab5$ ./fork1
Hello World
harsh@Ubuntu:~/Desktop/OS Lab Course/lab5$ Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

Lab 6: Pipe System Call

A user programme can interact with the operating system using a system call. A number of services are requested by the programme, and the OS replies by launching a number of system calls to fulfill the request. A system call can be written in high-level languages like C or Pascal or in assembly code. If a high-level language is employed, the operating system may directly invoke system calls, which are predefined functions.

The operating system's features are linked to user programmes using the Application Programme Interface (API). It serves as a conduit between a process and the operating system, enabling user-level programmes to ask for operating system services. System calls are the only means of communication with the kernel system. Any programme that uses resources must use system calls.

One of the most prevalent system calls is the pipe system call. Pipe is a communication medium between two or more related or interrelated processes. It can be either within one process or a communication between the child and the parent processes. Communication can also be multi-level such as communication between the parent, the child, and the grand-child, etc. Communication is achieved by one process writing into the pipe and other reading from the pipe. To achieve the pipe system call, create two files, one to write into the file and another to read from the file.

Pipe mechanism can be viewed with a real-time scenario such as filling water with the pipe into some container, say a bucket, and someone retrieving it, say with a mug. The filling process is nothing but writing into the pipe and the reading process is nothing but retrieving from the pipe. This implies that one output (water) is input for the other (bucket).

Pipe() call

The pipe() system function is used to open file descriptors, which are used to communicate between different Linux processes. It creates a pipe, a unidirectional data channel that can be used for inter-process communication. The array ***pipefd*** is used to return two file descriptors referring to the ends of the pipe. ***pipefd[0]*** refers to the read end of the pipe. ***pipefd[1]*** refers to the write end of the pipe. Data written to the write end of the pipe is buffered by the kernel until it is read from the rear end of the pipe.

Syntax:

int pipe(int pipefd[2]);

Here, the pipe() function creates a unidirectional data channel for inter-process communication. You pass in an int (Integer) type array *pipefd* consisting of 2 array elements to the function *pipe()*. Then the pipe() function creates two file descriptors in the *pipefd* array.

The first element of the *pipefd* array, *pipefd[0]* is used for reading data from the pipe. The second element of the *pipefd* array, *pipefd[1]* is used for writing data to the pipe. On success, the *pipe()* function returns 0. If an error occurs during pipe initialization, then the *pipe()* function returns -1. The pipe() function is defined in the header *unistd.h*. In order to use the *pipe()* function in C program, the header *unistd.h* must be included.

Implementation

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(void)
{
        int pipefds[2];
        if(pipe(pipefds) == -1)
        {
                perror("pipe");
                exit(EXIT_FAILURE);
        }
        printf("Read File Descriptor Value: %d\n", pipefds[0]);
        printf("Write File Descriptor Value: %d\n", pipefds[1]);
        return EXIT_SUCCESS;
}
```

```
harsh@Ubuntu:~/Desktop/OS Lab Course/lab6$ gcc pipe.c -o pipe
harsh@Ubuntu:~/Desktop/OS Lab Course/lab6$ ./pipe
Read File Descriptor Value: 3
Write File Descriptor Value: 4
```

The following example illustrates the use of pipe for inter-process communication. A PIN is sent from the child process to the parent process using a pipe. It is then read the PIN from the pipe in the parent process and printed it from the parent process. The code is shown below:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
int main(void)
{
        int pipefds[2];
        char buffer[5];
        if (pipe(pipefds) == -1)
        {
                perror("pipe");
                exit(EXIT_FAILURE);
        }
        char *pin = "4128\0";
        printf("Writing PIN to pipe\n");
        write(pipefds[1], pin, 5);
        printf("Done\n\n");
        printf("Reading PIN from pipe\n");
```

```
        read(pipefds[0], buffer, 5);
        printf("Done\n\n");
        printf("PIN from pipe: %s\n", buffer);
        return EXIT_SUCCESS;
}
```

```
harsh@Ubuntu:~/Desktop/OS Lab Course/lab6$ gcc pipe.c -o pipe
harsh@Ubuntu:~/Desktop/OS Lab Course/lab6$ ./pipe
Writing PIN to pipe
Done

Reading PIN from pipe
Done

PIN from pipe: 4128
```

Lab 7: CPU Scheduling

1. First Come First Serve (FCFS)

```
import java.util.Scanner;

      public class FCFS {
          public static void main(String[] args) {
              Scanner input = new Scanner(System.in);

              // Take the number of processes as input
              System.out.print("Enter number of processes: ");
              int n = input.nextInt();
              System.out.println();

              // Initialize arrays to store arrival time, burst time, waiting
      time, turn around time, and completion status of each process
              int[] at = new int[n];
              int[] bt = new int[n];
              int[] tat = new int[n];
              int[] wt = new int[n];
              boolean[] completed = new boolean[n];

              // Take the arrival time and burst time of each process as
      input
              for(int i = 0; i < n; i ++) {
                  System.out.print("Enter Arrival Time of P" + (i + 1) + ":
      ");
                  at[i] = input.nextInt();
                  System.out.print("Enter Burt Time of P" + (i + 1) + ": ");
                  bt[i] = input.nextInt();
              }

              int currentTime = 0;
              int completedProcesses = 0;

              // Loop until all processes have been completed
              while(completedProcesses < n) {
                  int Job = -1;
                  int shortestArrivalTime = Integer.MAX_VALUE;

                  // Find the process with the shortest arrival time that has
      arrived and has not yet been completed
                  for(int i = 0; i < n; i ++) {
                      if(at[i] <= currentTime && completed[i] == false &&
      at[i] < shortestArrivalTime) {
                          Job = i;
                          shortestArrivalTime = at[i];
```

```
                    }
                }

                // If there are no such processes, increment the current
      time by 1
                if(Job == -1) {
                    currentTime ++;
                }
                // If there is such a process, calculate the waiting time
      and turn around time of the process, update the current time, mark the
      process as completed, and increment the number of completed processes
                else {
                    wt[Job] = currentTime - at[Job];
                    tat[Job] = bt[Job] + wt[Job];
                    currentTime = currentTime + bt[Job];
                    completed[Job] = true;
                    completedProcesses ++;
                }
            }

            // Calculate the average waiting time and average turn around
      time
            double avgwt = 0.0;
            double avgtat = 0.0;

      //        System.out.println("\nP \tAT\tBT\tET\tWT\tTT");
            for(int i = 0; i < n; i ++) {
      //            System.out.printf("P%d\t%d\t%d\t%d\t%d\t%d\n", i+1,
      at[i], bt[i], tat[i] + bt[i], tat[i], wt[i]);
                avgwt = avgwt + wt[i];
                avgtat = avgtat + tat[i];
            }

            avgwt = avgwt / n;
            avgtat = avgtat / n;

            // Print the average waiting time and average turn around time
            System.out.println("\nAverage Waiting Time is " + avgwt);
            System.out.println("Average Turn Around Time is " + avgtat);

            input.close();
        }
}
```

Output:

```
Enter number of processes: 5

Enter Arrival Time of P1: 3
Enter Burt Time of P1: 4
Enter Arrival Time of P2: 5
Enter Burt Time of P2: 3
Enter Arrival Time of P3: 0
Enter Burt Time of P3: 2
Enter Arrival Time of P4: 5
Enter Burt Time of P4: 1
Enter Arrival Time of P5: 4
Enter Burt Time of P5: 3

Average Waiting Time is 3.2
Average Turn Around Time is 5.8
```

2. Shortest Job First (SJF)

```
import java.util.Scanner;


      public class SJF {
          public static void main(String[] args) {
              Scanner input = new Scanner(System.in);

              // Take the number of processes as input
              System.out.print("Enter number of processes: ");
              int n = input.nextInt();
              System.out.println();

              // Initialize arrays to store arrival time, burst time, waiting
      time, turn around time, and completion status of each process
              int[] at = new int[n];
              int[] bt = new int[n];
              int[] tat = new int[n];
              int[] wt = new int[n];
              boolean[] completed = new boolean[n];

              // Take the arrival time and burst time of each process as
      input
              for(int i = 0; i < n; i ++) {
                  System.out.print("Enter Arrival Time of P" + (i + 1) + ":
      ");
                  at[i] = input.nextInt();
                  System.out.print("Enter Burt Time of P" + (i + 1) + ": ");
                  bt[i] = input.nextInt();
```

```
        }

        int currentTime = 0;
        int completedProcesses = 0;

        // Loop until all processes have been completed
        while(completedProcesses < n) {
            int shortestJob = -1;
            int shortestBurstTime = Integer.MAX_VALUE;

            // Find the process with the shortest burst time that has
arrived and has not yet been completed
            for(int i = 0; i < n; i++) {
                if(at[i] <= currentTime && completed[i] == false &&
bt[i] < shortestBurstTime) {
                    shortestJob = i;
                    shortestBurstTime = bt[i];
                }
            }

            // If there are no such processes, increment the current
time by 1
            if(shortestJob == -1) {
                currentTime ++;
            }

            // If there is such a process
            // Calculate the waiting time and turn around time of the
process
            // Udate the current time
            // Mark the process as completed
            // Increment the number of completed processes
            else {
                wt[shortestJob] = currentTime - at[shortestJob];
                tat[shortestJob] = bt[shortestJob] + wt[shortestJob];
                currentTime = currentTime + bt[shortestJob];
                completed[shortestJob] = true;
                completedProcesses++;
            }
        }

        // Calculate the average waiting time and average turn around
time
        double avgwt = 0.0;
        double avgtat = 0.0;
        for(int i = 0; i < n; i ++) {
            avgwt = avgwt + wt[i];
            avgtat = avgtat + tat[i];
        }
```

```
            avgwt = avgwt / n;
            avgtat = avgtat / n;

            // Print the average waiting time and average turn around time
            System.out.println("\nAverage Waiting Time is " + avgwt);
            System.out.println("Average Turn Around Time is " + avgtat);

            input.close();
        }
}
```

Output:

```
Enter number of processes: 5

Enter Arrival Time of P1: 3
Enter Burt Time of P1: 1
Enter Arrival Time of P2: 1
Enter Burt Time of P2: 4
Enter Arrival Time of P3: 4
Enter Burt Time of P3: 2
Enter Arrival Time of P4: 0
Enter Burt Time of P4: 6
Enter Arrival Time of P5: 2
Enter Burt Time of P5: 3

Average Waiting Time is 4.8
Average Turn Around Time is 8.0
```

3. Shortest Remaining Time First (SRTF)

```
import java.util.Scanner;

      public class SRTF {
          public static void main(String[] args) {
              Scanner input = new Scanner(System.in);

              // Take the number of processes as input
              System.out.print("Enter number of processes: ");
              int n = input.nextInt();
              System.out.println();

              // Initialize arrays to store arrival time, burst time,
      remaining time, completion time, waiting time, turn around time, and
      completion status of each process
              int[] at = new int[n];
              int[] bt = new int[n];
```

```
            int[] rt = new int[n];
            int[] ct = new int[n];
            int[] tat = new int[n];
            int[] wt = new int[n];
            boolean[] completed = new boolean[n];

            // Take the arrival time and burst time of each process as
        input and initialize the remaining time of each process to its burst
        time
            for(int i = 0; i < n; i ++) {
                System.out.print("Enter Arrival Time of P" + (i + 1) + ":
        ");
                at[i] = input.nextInt();
                System.out.print("Enter Burst Time of P" + (i + 1) + ": ");
                bt[i] = input.nextInt();
                rt[i] = bt[i];
            }

            int currentTime = 0;
            int completedProcesses = 0;

            // Loop until all processes have been completed
            while(completedProcesses < n) {
                int shortestJob = -1;
                int shortestRemainingTime = Integer.MAX_VALUE;

                // Find the process with the shortest remaining time that
        has arrived and has not yet been completed
                for(int i = 0; i < n; i ++) {
                    if(at[i] <= currentTime && completed[i] == false &&
        rt[i] < shortestRemainingTime) {
                        shortestJob = i;
                        shortestRemainingTime = rt[i];
                    }
                }
                // If there are no such processes, increment the current
        time by 1
                if(shortestJob == -1) {
                    currentTime ++;
                }
                // If there is such a process
                else {
                    if(shortestRemainingTime == 1) {
                        ct[shortestJob] = currentTime + 1;
                        completed[shortestJob] = true;
                        completedProcesses ++;
                    }
                    rt[shortestJob] = rt[shortestJob] - 1;
                    currentTime ++;
```

```
                }
            }

            // Calculate the waiting time and turn around time of each
    process
            for(int i = 0; i < n; i ++) {
                tat[i] = ct[i] - at[i];
                wt[i] = tat[i] - bt[i];
            }

            // Calculate the average waiting time and average turn around
    time
            double avgwt = 0.0;
            double avgtat = 0.0;
            for(int i = 0; i < n; i ++) {
                avgwt = avgwt + wt[i];
                avgtat = avgtat + tat[i];
            }

            avgwt = avgwt / n;
            avgtat = avgtat / n;

            // Print the average waiting time and average turn around time
            System.out.println("\nAverage Waiting Time is " + avgwt);
            System.out.println("Average Turn Around Time is " + avgtat);

            input.close();
        }
}
```

Output:

```
Enter number of processes: 5

Enter Arrival Time of P1: 3
Enter Burst Time of P1: 1
Enter Arrival Time of P2: 1
Enter Burst Time of P2: 4
Enter Arrival Time of P3: 4
Enter Burst Time of P3: 2
Enter Arrival Time of P4: 0
Enter Burst Time of P4: 6
Enter Arrival Time of P5: 2
Enter Burst Time of P5: 3

Average Waiting Time is 3.8
Average Turn Around Time is 7.0
```

4. Round Robin

```
import java.util.*;

      public class Round_Robbin {
          public static void main(String[] args) {
              Scanner input = new Scanner(System.in);

              // Input number of processes
              System.out.print("Enter the number of processes: ");
              int n = input.nextInt();

              // Creating an "Array of Arraylist" to store data of each
      process
              ArrayList<int[]> processes = new ArrayList<int[]>();

              // Input Arrival Time & Burst Time
              for (int i = 0; i < n; i++) {
                  System.out.printf("Enter arrival time for process %d: ",
      i+1);
                  int at = input.nextInt();
                  System.out.printf("Enter burst time for process %d: ",
      i+1);
                  int bt = input.nextInt();
                  processes.add(new int[] {at, bt, bt, 0, 0, 0});
              }

              // Input Time Quantum
              System.out.print("Enter time quantum: ");
              int quantum = input.nextInt();

              // Declarations
              int time = 0;
              double averageWaitingTime = 0;
              double averageTurnaroundTime = 0;

              // Sorting list according to Arrival time
              processes.sort(Comparator.comparingInt(process -> process[0]));

              // Creating a Ready Queue
              Queue <Integer> readyQueue = new LinkedList<>();
              readyQueue.add(0);

              while (!readyQueue.isEmpty()) {

                  // Ready the first ready process
                  int i = readyQueue.poll();
```

```
                // If Remaining burst time of process is less than or equal
        to Time Quantum
                if (processes.get(i)[2] <= quantum) {
                    time += processes.get(i)[2];
                    processes.get(i)[2] = 0;
                    processes.get(i)[3] = time;
                    processes.get(i)[4] = processes.get(i)[3] -
        processes.get(i)[0];
                    processes.get(i)[5] = processes.get(i)[4] -
        processes.get(i)[1];
                }

                // If Remaining burst time of process is greater than Time
        Quantum
                else {
                    time += quantum;
                    processes.get(i)[2] -= quantum;
                    // Create a new ArrayList of integers to store the
        indices of the processes that are ready to be executed
                    ArrayList<Integer> temp = new ArrayList<>();

                    // Loop through all processes and check if they are
        ready to be executed based on their arrival time and remaining burst
        time
                    for (int j = 0; j < n; j++) {
                        if (processes.get(j)[0] <= time && j != i &&
        !readyQueue.contains(j) && processes.get(j)[2] != 0) {
                            // If a process is ready to be executed, add
        its index to the temporary ArrayList
                            temp.add(j);
                        }
                    }
                    // Add all the processes that are ready to be executed
        to the queue
                    readyQueue.addAll(temp);

                    // Add the current process to the queue
                    readyQueue.offer(i);
                }
            }

            // Calculating Average Waiting Time & Average Turnaround Time
            System.out.println("\nP \tAT\tBT\tET\tWT\tTT");
            for (int i = 0; i < n; i++) {
                System.out.printf("P%d\t%d\t%d\t%d\t%d\t%d\n", i+1,
        processes.get(i)[0], processes.get(i)[1], processes.get(i)[3],
        processes.get(i)[4], processes.get(i)[5]);
                averageWaitingTime += processes.get(i)[5];
                averageTurnaroundTime += processes.get(i)[4];
```

```
            }

            averageWaitingTime /= n;
            averageTurnaroundTime /= n;

            // Printing Final Results
            System.out.printf("\nAverage Waiting Time: %.2f\n",
    averageWaitingTime);
            System.out.printf("Average Turnaround Time: %.2f\n",
    averageTurnaroundTime);

            input.close();

        }
    }
```

Output:

```
Enter the number of processes: 5
Enter arrival time for process 1: 0
Enter burst time for process 1: 5
Enter arrival time for process 2: 1
Enter burst time for process 2: 3
Enter arrival time for process 3: 2
Enter burst time for process 3: 1
Enter arrival time for process 4: 3
Enter burst time for process 4: 2
Enter arrival time for process 5: 4
Enter burst time for process 5: 3
Enter time quantum: 2

P   AT  BT  ET  WT  TT
P1  0   5   13  13  8
P2  1   3   12  11  8
P3  2   1   5   3   2
P4  3   2   9   6   4
P5  4   3   14  10  7

Average Waiting Time: 5.80
Average Turnaround Time: 8.60
```

Lab 8: Disk Scheduling

1. First Come First Serve (FCFS)

```
import java.util.*;

public class FCFS {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter number of requests: ");
      int n = sc.nextInt();

      int[] requests = new int[n];

      for (int i=1 ; i<=n ; i++) {
         System.out.printf("Enter value of P%d : ", i);
         requests[i-1] = sc.nextInt();
      }
      System.out.println(Arrays.toString(requests));

      System.out.print("Enter Head value: ");
      int head = sc.nextInt();

      int seekTime = 0;

      for (int i=0 ; i<n ; i++) {
         if (head > requests[i]) {
            seekTime = seekTime - (requests[i] - head);
         }
         else {
            seekTime = seekTime + (requests[i] - head);
         }
         head = requests[i];
      }

      System.out.println("Seek Time : " + seekTime);

      sc.close();
   }
}
```

Output:

```
Enter number of requests: 7
Enter value of P1 : 82
Enter value of P2 : 170
Enter value of P3 : 43
Enter value of P4 : 140
Enter value of P5 : 24
Enter value of P6 : 16
Enter value of P7 : 190
[82, 170, 43, 140, 24, 16, 190]
Enter Head value: 50
Seek Time : 642
```

2. Shortest Seek Time First (SSTF)

```
import java.util.Scanner;

public class SSTF {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter number of requests: ");
        int n = input.nextInt();
        int[] requests = new int[n];

        for(int i = 0; i < n; i ++) {
            System.out.print("Enter Request " + (i + 1) + ": ");
            requests[i] = input.nextInt();
        }

        System.out.print("Enter Head location: ");
        int head = input.nextInt();

        int seekTime = 0;
        boolean[] completed = new boolean[n];

        // Main Programm
        for (int i=0 ; i<n ; i++) {
            int[] difference = findSeekTime(requests, head, completed);
            int index = findIndex(difference);
            seekTime += difference[index];
            completed[index] = true;
```

```
            head = requests[index];
        }

        System.out.println("Total Seek Time for serving all requests : " + seekTime);
        input.close();
    }

    public static int findDifference(int a, int b) {
        if (a > b) { return a-b; }
        else { return b-a; }
    }

    public static int[] findSeekTime(int[] requests, int head, boolean[] completed) {
        int[] difference = new int[requests.length];

        for (int i=0 ; i<requests.length ; i++) {
            if (!completed[i]) {
                difference[i] = findDifference(head, requests[i]);
            } else {
                difference[i] = Integer.MAX_VALUE;
            }
        }
        return difference;
    }

    public static int findMin (int[] array) {
        int min = Integer.MAX_VALUE;
        for (int i=0 ; i<array.length ; i++) {
            if (array[i] < min) {
                min = array[i];
            }
        }
        return min;
    }

    public static int findIndex (int[] array) {
        int i = 0;
        int index = -1;
        int min = findMin(array);
        while(i < array.length) {
            if(array[i] == min) {
                index = i;
                break;
            }
            i++;
        }
```

```
        return index;
    }
}
```

Output:

```
Enter number of requests: 7
Enter Request 1: 82
Enter Request 2: 170
Enter Request 3: 43
Enter Request 4: 140
Enter Request 5: 24
Enter Request 6: 16
Enter Request 7: 190
Enter Head location: 50
Total Seek Time for serving all requests : 208
```

3. SCAN / Elevator Algorithm

```
import java.util.Scanner;

public class SCAN {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of requests: ");
        int n = sc.nextInt();
        int[] requests = new int[n];

        for(int i = 0; i < n; i ++) {
            System.out.print("Enter Request " + (i + 1) + ": ");
            requests[i] = sc.nextInt();
        }

        System.out.print("Enter Head Location: ");
        int head = sc.nextInt();

        System.out.print("Enter Disk Size: ");
        int diskSize = sc.nextInt();

        System.out.print("\nEnter Direction\n1. Towards Lesser Requests\n2. Towards Greater Requests\n-> ");
        int direction = sc.nextInt();
```

```
int seekTime = 0;
boolean[] completed = new boolean[n];
int Distance = 0;

if (direction == 1) {
   while(head >= 0) {
      for(int i = 0; i < n; i ++) {
         if(requests[i] == head && completed[i] == false) {
            seekTime = seekTime + Distance;
            Distance = 0;
         }
      }
      Distance ++;
      head --;
   }
   while(head < diskSize) {
      for(int i = 0; i < n; i ++) {
         if(requests[i] == head) {
            seekTime = seekTime + Distance;
            completed[i] = true;
            Distance = 0;
         }
      }
      Distance ++;
      head ++;
   }
}

else if (direction == 2) {
   while(head < diskSize) {
      for(int i = 0; i < n; i ++) {
         if(requests[i] == head) {
            seekTime = seekTime + Distance;
            completed[i] = true;
            Distance = 0;
         }
      }
      Distance ++;
      head ++;
   }
   while(head >= 0) {
      for(int i = 0; i < n; i ++) {
         if(requests[i] == head && completed[i] == false) {
            seekTime = seekTime + Distance;
            Distance = 0;
         }
```

```
            }
            Distance ++;
            head --;
        }
    }
    System.out.println("\nTotal Seek Time for serving all requests: " + seekTime);
    sc.close();
  }
}
```

Output:

```
Enter number of requests: 7
Enter Request 1: 82
Enter Request 2: 170
Enter Request 3: 43
Enter Request 4: 140
Enter Request 5: 24
Enter Request 6: 16
Enter Request 7: 190
Enter Head Location: 50
Enter Disk Size: 199

Enter Direction
1. Towards Lesser Requests
2. Towards Greater Requests
-> 2

Total Seek Time for serving all requests: 332
```

4. Circular - SCAN (CSCAN)

```
import java.util.Scanner;

public class CSCAN {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    System.out.print("Enter number of requests: ");
    int n = sc.nextInt();
    int[] requests = new int[n];

    for(int i = 0; i < n; i ++) {
      System.out.print("Enter Request " + (i + 1) + ": ");
```

```
            requests[i] = sc.nextInt();
        }

        System.out.print("Enter Head Location: ");
        int head = sc.nextInt();

        System.out.print("Enter Disk Size: ");
        int diskSize = sc.nextInt();

        int initialHead = head;
        int seekTime = 0;
        int Distance = 0;

        while(head < diskSize) {
            for(int i = 0; i < n; i ++) {
                if(requests[i] == head) {
                    seekTime = seekTime + Distance;
                    Distance = 0;
                }
            }
            Distance ++;
            head ++;
        }

        seekTime = seekTime + diskSize;
        head = 0;

        while(head <= initialHead) {
            for(int i = 0; i < n; i ++) {
                if(requests[i] == head) {
                    seekTime = seekTime + Distance;
                    Distance = 0;
                }
            }
            Distance ++;
            head ++;
        }
        System.out.println("\nTotal Seek Time for serving all requests is " + seekTime);
        sc.close();
    }
}
```

Output:

```
Enter number of requests: 7
Enter Request 1: 82
Enter Request 2: 170
Enter Request 3: 43
Enter Request 4: 140
Enter Request 5: 24
Enter Request 6: 16
Enter Request 7: 190
Enter Head Location: 50
Enter Disk Size: 199

Total Seek Time for serving all requests is 391
```

5. LOOK

```
import java.util.Scanner;

public class LOOK {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter number of requests: ");
      int n = sc.nextInt();
      int[] requests = new int[n];

      for(int i = 0; i < n; i ++) {
         System.out.print("Enter Request " + (i + 1) + ": ");
         requests[i] = sc.nextInt();
      }

      System.out.print("Enter Head Location: ");
      int head = sc.nextInt();
      int seekTime = 0;
      boolean[] completed = new boolean[n];
      int Distance = 0;
      int upperBound = Integer.MIN_VALUE;
      int lowerBound = Integer.MAX_VALUE;

      for(int i = 0; i < n; i ++) {
         if(requests[i] > upperBound) {
            upperBound = requests[i];
```

```
        }
        if(requests[i] < lowerBound) {
            lowerBound = requests[i];
        }
    }

    while(head < upperBound) {
        for(int i = 0; i < n; i ++) {
            if(requests[i] == head) {
                seekTime = seekTime + Distance;
                completed[i] = true;
                Distance = 0;
            }
        }
        Distance ++;
        head ++;
    }

    while(head >= lowerBound) {
        for(int i = 0; i < n; i ++) {
            if(requests[i] == head && completed[i] == false) {
                seekTime = seekTime + Distance;
                Distance = 0;
            }
        }
        Distance ++;
        head --;
    }
    System.out.println("Total Seek Time for serving all requests is " + seekTime);
    sc.close();
  }
}
```

Output:

```
Enter number of requests: 7
Enter Request 1: 82
Enter Request 2: 170
Enter Request 3: 43
Enter Request 4: 140
Enter Request 5: 24
Enter Request 6: 16
Enter Request 7: 190
Enter Head Location: 50
Total Seek Time for serving all requests is 314
```

6. Circular – LOOK (CLOOK)

```
import java.util.Scanner;

public class CLOOK {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter number of requests: ");
      int n = sc.nextInt();
      int[] requests = new int[n];

      for(int i = 0; i < n; i ++) {
         System.out.print("Enter Request " + (i + 1) + ": ");
         requests[i] = sc.nextInt();
      }

      System.out.print("Enter Head Location: ");
      int head = sc.nextInt();
      int seekTime = 0;
      int Distance = 0;
      int upperBound = Integer.MIN_VALUE;
      int lowerBound = Integer.MAX_VALUE;
      int initialHead = head;

      for(int i = 0; i < n; i ++) {
         if(requests[i] > upperBound) {
            upperBound = requests[i];
         }
         if(requests[i] < lowerBound) {
            lowerBound = requests[i];
         }
      }

      while(head < upperBound) {
         for(int i = 0; i < n; i ++) {
            if(requests[i] == head) {
               seekTime = seekTime + Distance;
               Distance = 0;
            }
         }
         Distance ++;
         head ++;
      }
```

```
        head = lowerBound;


        seekTime = seekTime + upperBound - lowerBound;

        while(head < initialHead) {
            for(int i = 0; i < n; i ++) {
                if(requests[i] == head) {
                    seekTime = seekTime + Distance;
                    Distance = 0;
                }
            }
            Distance ++;
            head ++;
        }
        System.out.println("\nTotal Seek Time for serving all requests: " + seekTime);
        sc.close();
    }
}
```

Output:

```
Enter number of requests: 7
Enter Request 1: 82
Enter Request 2: 170
Enter Request 3: 43
Enter Request 4: 140
Enter Request 5: 24
Enter Request 6: 16
Enter Request 7: 190
Enter Head Location: 50

Total Seek Time for serving all requests: 341
```

Lab 9: Deadlock and Concurrency

1. Producer Consumer

```
import java.util.*;
public class ProducerConsumer {
    static Scanner sc = new Scanner(System.in);
    static int mutex = 1;
    static int pos = -1;
    static int n = 3;
    static String item;
    static Stack<String> newBuffer = new Stack<String>();

    public static int wait(int s) {
        while (s != 1) ;
        return (--s);
    }
    public static int signal(int s) {
        return (++s);
    }
    public static void producer () {
        mutex = wait(mutex);
        pos = signal(pos);
        if (pos < n) {
            System.out.print("Enter Item to Produce: ");
            String item = sc.next();
            System.out.println("Produced item '" + item + "'");
            newBuffer.push(item);
        }
        mutex = signal(mutex);
    }
    public static void consumer () {
        mutex = wait(mutex);
        pos--;
        if (pos >= -1) {
```

```
            item = newBuffer.pop();
            System.out.println("Consumed item '" + item + "'");
        }
        mutex = signal(mutex);
    }
    public static void display () {
        if (newBuffer.size() == 0) {
            System.out.print("Buffer -> EMPTY");
        }
        else {
            System.out.print("Buffer -> ");
            for (String i : newBuffer) {
                System.out.print(i + " ");
            }
            System.out.println();
        }
    }
    public static void main(String[] args) {
        System.out.print("Enter Buffer size : ");
        n = sc.nextInt();

        System.out.println("\n1. Producer\n2. Consumer\n3. Display Buffer\n4. Exit");
        boolean loop = true;
        while (loop) {
            System.out.print("\nEnter your choice: ");
            int choice = sc.nextInt();

            switch (choice) {
                case (1) -> {
                    if (mutex == 1 && (pos+1) < n) {
                        producer();
                    } else {
                        System.out.println("Buffer is full, There's no space to Produce!");
                    }
                }
```

```
            case (2) -> {
                if (mutex == 1 && pos >= 0) {
                    consumer();
                } else {
                    System.out.println("Buffer is empty, There's nothing to Consume!");
                }
            }
            case (3) -> display();
            case (4) -> {
                System.out.println("\nThank You!");
                loop = false;
            }
            default -> System.out.println("Please Enter correct Choice");
        }
    }
  }
}
```

Output:

```
Enter Buffer size : 3

1. Producer
2. Consumer
3. Display Buffer
4. Exit

Enter your choice: 1
Enter Item to Produce: 21
Produced item '21'

Enter your choice: 1
Enter Item to Produce: 56
Produced item '56'

Enter your choice: 1
Enter Item to Produce: 85
Produced item '85'
```

```
Enter your choice: 1
Buffer is full, There's no space to Produce!
Enter your choice: 3
Buffer -> 21 56 85

Enter your choice: 2
Consumed item '85'

Enter your choice: 2
Consumed item '56'

Enter your choice: 2
Consumed item '21'
```

```
Enter your choice: 3
Buffer -> EMPTY
Enter your choice: 2
Buffer is empty, There's nothing to Consume!

Enter your choice: 4

Thank You!
```

2. Bankers Algorithm

```
#include<stdio.h>
#include<stdbool.h>

int P = 5;
int R = 3;

//finding needs of each process
void calculateNeed(int need[P][R], int max[P][R], int allot[P][R]) {
   for (int i = 0 ; i < P ; i++) {
      for (int j = 0 ; j < R ; j++) {
         need[i][j] = max[i][j] - allot[i][j];
      }

   }
}

// Function to find the system is in safe state or not
bool isSafe(int processes[], int avail[], int max[P][R], int allot[P][R]) {
```

```
int need[P][R];
calculateNeed(need, max, allot);

bool finish[5] = {0,0,0,0,0};
bool found;
int safeSeq[P];

int work[R];
for (int i = 0; i < R ; i++){
   work[i] = avail[i];
}

int count = 0;
while (count < P) {
   found = false;
   for (int i = 0; i < P; i++) {
      if (finish[i] == 0) {
         int j;
         for (j = 0; j < R; j++) {
            if (need[i][j] > work[j]) {
               break;
            }
         }

         if (j == R) {

            for (int k = 0 ; k < R ; k++) {
               work[k] += allot[i][k];
            }

            safeSeq[count++] = i;
            finish[i] = 1;
            found = true;
         }
      }
   }
}

if (found == false) {
   printf("System is not in safe state");
   return false;
}

printf("System is in safe state.\n");
printf("Safe sequence is: ");
for (int i = 0; i < P ; i++) {
```

```
      printf("%d ", safeSeq[i]);
    }
    return true;
}

void main() {
    int processes[] = {0, 1, 2, 3, 4};

    // Available matrix
    int avail[] = {3, 3, 2};

    // max matrix
    int max[5][3] = {
    {7, 5, 3},
    {3, 2, 2},
    {9, 0, 2},
    {2, 2, 2},
    {4, 3, 3}};

    // allotted matrix
    int allot[5][3] = {
    {0, 1, 0},
    {2, 0, 0},
    {3, 0, 2},
    {2, 1, 1},
    {0, 0, 2}};

    isSafe(processes, avail, max, allot);
}
```

Output:

```
System is in safe state.
Safe sequence is: 1 3 4 0 2
```

Lab 10: Page Replacement Algorithms

1. First In First Out

```
import java.util.Scanner;
public class FIFO {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        // Inputs
      System.out.print("Enter Page frame: ");
        int page = input.nextInt();

        System.out.print("Enter number of entries in queue: ");
        int entry = input.nextInt();

        int[] frame = new int[page];
        int[] entries = new int[entry];
        for (int i = 0; i < entry; i++) {
            System.out.print("Enter value of entry " + (i + 1) + ": ");
            entries[i] = input.nextInt();
        }
        for (int i = 0; i < page; i++) {
            frame[i] = -1;
        }
        int miss = 0;
        int next = 0;
        for (int i = 0; i < entry; i++) {
            int count = 0;
            for (int j = 0; j < page; j++) {
                if (frame[j] == entries[i]) {
                    break;
                }
                count++;
            }
            if (count == page) {
                miss++;
                frame[next] = entries[i];
                next = (next + 1) % page;
            }
        }
      System.out.println("Page Faults: " + miss);
      System.out.println("Page Hits: " + (entry -miss));
        input.close();
    }
}
```

Output:

```
Enter Page frame: 4
Enter number of entries in queue: 14
Enter value of entry 1: 7
Enter value of entry 2: 0
Enter value of entry 3: 1
Enter value of entry 4: 2
Enter value of entry 5: 0
Enter value of entry 6: 3
Enter value of entry 7: 0
Enter value of entry 8: 4
Enter value of entry 9: 2
Enter value of entry 10: 3
Enter value of entry 11: 0
Enter value of entry 12: 3
Enter value of entry 13: 2
Enter value of entry 14: 3
Page Faults: 7
Page Hits: 7
```

2. Least Recently Used

```
import java.util.Scanner;
public class LRU {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      // Inputs
      System.out.print("Enter Page frame: ");
      int page = input.nextInt();

            System.out.print("Enter number of entries in queue: ");
      int entry = input.nextInt();

      int[] frame = new int[page];
      int[] entries = new int[entry];
      for (int i = 0; i < entry; i++) {
         System.out.print("Enter value of entry " + (i + 1) + ": ");
         entries[i] = input.nextInt();
      }
      for (int i = 0; i < page; i++) {
         frame[i] = -1;
      }
      int miss = page;
      int[] used = new int[page];
      for (int i = 0; i < page; i++) {
         frame[i] = entries[i];
      }
      for (int i = page; i < entry; i++) {
         int count = 0;
         for (int j = 0; j < page; j++) {
            if (entries[i] == frame[j]) {
               for (int k = 0; k < page; k++) {
```

```
                    used[k]++;
                }
                used[j] = 0;
                break;
            }
            count++;
        }
        if (count == page) {
            for (int j = 0; j < page; j++) {
                used[j]++;
            }
            int max = Integer.MIN_VALUE;
            int maxIndex = 0;
            for (int j = 0; j < page; j++) {
                if (used[j] > max) {
                    max = used[j];
                    maxIndex = j;
                }
            }
            used[maxIndex] = 0;
            frame[maxIndex] = entries[i];
            miss++;
        }
    }
    System.out.println("Page Faults: " + miss);
    System.out.println("Page Hits: " + (entry -miss));
    input.close();
  }
}
```

Output:

```
Enter Page frame: 4
Enter number of entries in queue: 14
Enter value of entry 1: 7
Enter value of entry 2: 0
Enter value of entry 3: 1
Enter value of entry 4: 2
Enter value of entry 5: 0
Enter value of entry 6: 3
Enter value of entry 7: 0
Enter value of entry 8: 4
Enter value of entry 9: 2
Enter value of entry 10: 3
Enter value of entry 11: 0
Enter value of entry 12: 3
Enter value of entry 13: 2
Enter value of entry 14: 3
Page Faults: 6
Page Hits: 8
```

3. Optimal Algorithm

```
import java.util.Scanner;

public class Optimal {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);

      // Inputs
      System.out.print("Enter Page frame: ");
      int page = input.nextInt();

      System.out.print("Enter number of entries in queue: ");
      int entry = input.nextInt();

      int[] frame = new int[page];
      int[] entries = new int[entry];
      for (int i = 0; i < entry; i++) {
         System.out.print("Enter value of entry " + (i + 1) + ": ");
         entries[i] = input.nextInt();
      }
      for (int i = 0; i < page; i++) {
         frame[i] = -1;
      }
      int miss = page;
      for (int i = 0; i < page; i++) {
         frame[i] = entries[i];
      }
      for (int i = page; i < entry; i++) {
         int count = 0;
         for (int j = 0; j < page; j++) {
            if (frame[j] == entries[i]) {
               break;
            }
            count++;
         }
         if (count == page) {
            int[] use = new int[page];
            for (int j = i; j < entry; j++) {
               for (int k = 0; k < page; k++) {
                  if (frame[k] == entries[j]) {
                     use[k]++;
                  }
               }
            }
            int min = Integer.MAX_VALUE;
            int minIndex = 0;
            for (int j = 0; j < page; j++) {
               if (use[j] < min) {
                  min = use[j];
                  minIndex = j;
               }
            }
```

```
                frame[minIndex] = entries[i];
                miss++;
            }
        }
        System.out.println("Page Faults: " + miss);
        System.out.println("Page Hits: " + (entry -miss));
        input.close();
    }
}
```

Output:

```
Enter Page frame: 4
Enter number of entries in queue: 14
Enter value of entry 1: 7
Enter value of entry 2: 0
Enter value of entry 3: 1
Enter value of entry 4: 2
Enter value of entry 5: 0
Enter value of entry 6: 3
Enter value of entry 7: 0
Enter value of entry 8: 4
Enter value of entry 9: 2
Enter value of entry 10: 3
Enter value of entry 11: 0
Enter value of entry 12: 3
Enter value of entry 13: 2
Enter value of entry 14: 3
Page Faults: 6
Page Hits: 8
```

4. Least Frequently Used

```
public class LFU {
    public static int pageFaults(int n, int c, int[] pages) {
        int count = 0;
        List<Integer> v = new ArrayList<>();
        Map<Integer, Integer> mp = new HashMap<>();
        int i;
        for (i = 0; i <= n - 1; i++) {
            int index = v.indexOf(pages[i]);
            if (index == -1) {
                if (v.size() == c) {
                    mp.put(v.get(0), mp.get(v.get(0))-1);
                    v.remove(0);
                }
                v.add(pages[i]);
                mp.put(pages[i], mp.getOrDefault(pages[i], 0)+1);
                count++;
            } else {
                mp.put(pages[i], mp.get(pages[i])+1);
                v.remove(index);
                v.add(pages[i]);
            }
            int k = v.size() - 2;
            while (k > -1 && mp.get(v.get(k)) > mp.get(v.get(k + 1))) {
                Collections.swap(v, k, k+1);
                k--;
            }
        }
        return count;
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        // Inputs
        System.out.print("Enter Page frame: ");
        int c = input.nextInt();

        System.out.print("Enter number of entries in queue: ");
        int n = input.nextInt();

        int[] pages = new int[n];
        for (int i = 0; i < n; i++) {
            System.out.print("Enter value of entry " + (i + 1) + ": ");
            pages[i] = input.nextInt();
        }

        System.out.println("Page Faults = " + pageFaults(n, c, pages));
        System.out.println("Page Hits = " + (n - pageFaults(n, c, pages)));

        input.close();
    }
}
```

Output:

```
Enter Page frame: 4 14
Enter number of entries in queue: Enter value of entry 1: 7
Enter value of entry 2: 0
Enter value of entry 3: 1
Enter value of entry 4: 2
Enter value of entry 5: 0
Enter value of entry 6: 3
Enter value of entry 7: 0
Enter value of entry 8: 4
Enter value of entry 9: 2
Enter value of entry 10: 3
Enter value of entry 11: 0
Enter value of entry 12: 3
Enter value of entry 13: 2
Enter value of entry 14: 3
Page Faults = 6
Page Hits = 8
```