

OS LAB WEEK – 1

1. To study and execute the basic commands in UNIX

Ans.

A) File related commands

```
osprac@osprac-virtual-machine:~$ mkdir oslab
osprac@osprac-virtual-machine:~$ ls
Desktop Downloads oslab Public Templates
Documents Music Pictures snap Videos
osprac@osprac-virtual-machine:~$ pwd
/home/osprac
osprac@osprac-virtual-machine:~$ cd oslab
osprac@osprac-virtual-machine:~/oslab$ touch file.txt
osprac@osprac-virtual-machine:~/oslab$ cat >file.txt
hello cbit
osprac@osprac-virtual-machine:~/oslab$ cat file.txt
hello cbit
osprac@osprac-virtual-machine:~/oslab$ ls
file.txt
osprac@osprac-virtual-machine:~/oslab$ ls -lrt
total 4
-rw-rw-r-- 1 osprac osprac 11 Sep  7 14:35 file.txt
```

```
osprac@osprac-virtual-machine:~$ cd oslab
osprac@osprac-virtual-machine:~/oslab$ ls
file.txt
osprac@osprac-virtual-machine:~/oslab$ touch file1.txt
osprac@osprac-virtual-machine:~/oslab$ cat >file1.txt
hello world
osprac@osprac-virtual-machine:~/oslab$ cmp file.txt file1.txt
file.txt file1.txt differ: byte 7, line 1
osprac@osprac-virtual-machine:~/oslab$ rm file1.txt
osprac@osprac-virtual-machine:~/oslab$ ls
file.txt
```

```
osprac@osprac-virtual-machine:~$ cd oslab
osprac@osprac-virtual-machine:~/oslab$ ls
subfoldr
osprac@osprac-virtual-machine:~/oslab$ cd subfoldr
bash: cd: subfoldr: Not a directory
osprac@osprac-virtual-machine:~/oslab$ mv subfoldr ./file.txt
osprac@osprac-virtual-machine:~/oslab$ ls
file.txt
osprac@osprac-virtual-machine:~/oslab$ cat file.txt
hello cbit
osprac@osprac-virtual-machine:~/oslab$ mv file.txt ./file1.txt
osprac@osprac-virtual-machine:~/oslab$ ls
file1.txt
osprac@osprac-virtual-machine:~/oslab$ cat file1.txt
hello cbit
osprac@osprac-virtual-machine:~/oslab$ cp file1.txt file2.txt
osprac@osprac-virtual-machine:~/oslab$ cat file2.txt
hello cbit
osprac@osprac-virtual-machine:~/oslab$ echo hello world
hello world
```

```
osprac@osprac-virtual-machine:~/oslab$ paste file1.txt file2.txt
hello cbit      hello cbit
osprac@osprac-virtual-machine:~/oslab$ mkdir subfolder
osprac@osprac-virtual-machine:~/oslab$ ls
file1.txt  file2.txt  subfolder
osprac@osprac-virtual-machine:~/oslab$ rmdir subfolder
osprac@osprac-virtual-machine:~/oslab$ ls
file1.txt  file2.txt
osprac@osprac-virtual-machine:~/oslab$ gedit file1.txt
osprac@osprac-virtual-machine:~/oslab$ head file1.txt
hello cbit
how is the college
is it fine??
hhhhhhhhhhhhhhh
jjjjjjjjjjjjjjjjjjjjjjjjjj
kkkkkkkkkkkkk
rrrrrrrrrrrrrrr
hello cbit
how is the college
is it fine??
osprac@osprac-virtual-machine:~/oslab$ tail file1.txt
kkkkkkkkkkkkk
rrrrrrrrrrrrrrr
hhhhhhhhhhhhhhh
jjjjjjjjjjjjjjjjjjjjjjjj
kkkkkkkkkkkkk
rrrrrrrrrrrrrrr
hhhhhhhhhhhhhhh
jjjjjjjjjjjjjjjjjjjjjjjj
kkkkkkkkkkkkk
rrrrrrrrrrrrrrr
```

```
osprac@osprac-virtual-machine:~/oslab$ date
Wednesday 07 September 2022 03:00:51 PM IST
osprac@osprac-virtual-machine:~/oslab$ grep "hello cbit" file1.txt file2.txt
file1.txt:hello cbit
file1.txt:hello cbit
file1.txt:hello cbit
file2.txt:hello cbit
osprac@osprac-virtual-machine:~/oslab$ ls -lrt
total 8
-rwxrwxr-X 1 osprac osprac 11 Sep  7 14:48 file2.txt
-rwxrwxrwx 1 osprac osprac 619 Sep  7 14:55 file1.txt
osprac@osprac-virtual-machine:~/oslab$ chmod 777 file2.txt
osprac@osprac-virtual-machine:~/oslab$ ls -lrt
total 8
-rwxrwxrwx 1 osprac osprac 11 Sep  7 14:48 file2.txt
-rwxrwxrwx 1 osprac osprac 619 Sep  7 14:55 file1.txt
osprac@osprac-virtual-machine:~/oslab$ chmod 421 file1.txt
osprac@osprac-virtual-machine:~/oslab$ ls -lrt
total 8
-rwxrwxrwx 1 osprac osprac 11 Sep  7 14:48 file2.txt
-rw---w--x 1 osprac osprac 619 Sep  7 14:55 file1.txt
osprac@osprac-virtual-machine:~/oslab$
```

```
osprac@osprac-virtual-machine:~/oslab$ ls -lrt
total 8
-rwxrwxrwx 1 osprac osprac 11 Sep  7 14:48 file2.txt
-rw---w--x 1 osprac osprac 619 Sep  7 14:55 file1.txt
osprac@osprac-virtual-machine:~/oslab$ chown master file1.txt
chown: invalid user: 'master'
osprac@osprac-virtual-machine:~/oslab$ sudo chown root file1.txt
[sudo] password for osprac:
osprac@osprac-virtual-machine:~/oslab$ ls -lrt
total 8
-rwxrwxrwx 1 osprac osprac 11 Sep  7 14:48 file2.txt
-rw---w--x 1 root    osprac 619 Sep  7 14:55 file1.txt
```

MAN(1) Manual pager utils **MAN(1)**

NAME
man - an interface to the system reference manuals

SYNOPSIS
man [**man options**] [[**section**] **page** ...] ...
man -k [**apropos options**] **regexp** ...
man -K [**man options**] [**section**] **term** ...
man -f [**whatis options**] **page** ...
man -l [**man options**] **file** ...
man -w|-W [**man options**] **page** ...

DESCRIPTION
man is the system's manual pager. Each **page** argument given to **man** is normally the name of a program, utility or function. The **manual page** associated with each of these arguments is then found and displayed. A **section**, if provided, will direct **man** to look only in that **section** of the manual. The default action is to search in all of the available **sections** following a pre-defined order (see **DEFAULTS**), and to show only the first **page** found, even if **page** exists in several **sections**.

The table below shows the **section** numbers of the manual followed by the types of pages they contain.

1	Executable programs or shell commands
2	System calls (functions provided by the kernel)
3	Library calls (functions within program libraries)
4	Special files (usually found in /dev)
5	File formats and conventions, e.g. /etc/passwd
6	Games
7	Miscellaneous (including macro packages and conventions), e.g. man(7) , groff(7) , man-pages(7)
8	System administration commands (usually only for root)
9	Kernel routines [Non standard]

A **manual page** consists of several sections.

Conventional section names include **NAME**, **SYNOPSIS**, **CONFIGURATION**, **DESCRIPTION**, **OPTIONS**, **EXIT STATUS**, **RETURN VALUE**, **ERRORS**, **ENVIRONMENT**, **FILES**, **Versions**, **CONFORMING TO**, **NOTES**, **BUGS**, **EXAMPLE**, **AUTHORS**, and **SEE ALSO**.

The following conventions apply to the **SYNOPSIS** section and can be used as a guide in other sections.

bold text	type exactly as shown.
italic text	replace with appropriate argument.
[-abc]	any or all arguments within [] are optional.
-a -b	options delimited by cannot be used together.
argument ...	argument is repeatable.
[expression] ...	entire expression within [] is repeatable.

Exact rendering may vary depending on the output device. For instance, **man** will usually not be able to render italics when running in a terminal, and will typically use underlined or coloured text instead.

```
Manual page man(1) line 1 (press h for help or q to quit)
```

B) Process related commands

Top command:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
7727	osprac	20	0	3259680	418872	202744	S	5.6	5.2	1:26.24	firefox
4757	osprac	20	0	595648	75812	61556	S	5.3	0.9	0:17.20	gnome-terminal
1108	osprac	20	0	5123232	331072	148400	S	2.7	4.1	2:09.79	gnome-shell
9388	osprac	20	0	2685780	151312	97600	S	1.7	1.9	0:16.93	Isolated Web Co
777	message+	20	0	10840	5988	3892	S	0.7	0.1	0:06.72	dbus-daemon
39	root	20	0	0	0	0	S	0.3	0.0	0:00.30	kcompactd0
330	root	20	0	0	0	0	S	0.3	0.0	0:00.87	jbd2/sda3-8
540	systemd+	20	0	14824	6204	5404	S	0.3	0.1	0:04.46	systemd-oomd
1202	root	20	0	250856	8700	7560	S	0.3	0.1	0:00.05	upowerd
4790	root	20	0	0	0	0	I	0.3	0.0	0:02.23	kworker/u256:1-writeback
9571	osprac	20	0	2401844	61696	50028	S	0.3	0.8	0:00.14	Web Content
9862	root	20	0	0	0	0	I	0.3	0.0	0:00.20	kworker/2:0-mm_percpu_wq
1	root	20	0	166564	11744	8256	S	0.0	0.1	0:04.03	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_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_rude_
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_trace
13	root	20	0	0	0	0	S	0.0	0.0	0:00.19	ksoftirqd/0
14	root	20	0	0	0	0	I	0.0	0.0	0:06.26	rcu_sched
15	root	rt	0	0	0	0	S	0.0	0.0	0:00.05	migration/0
16	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
19	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/1
20	root	rt	0	0	0	0	S	0.0	0.0	0:00.70	migration/1
21	root	20	0	0	0	0	S	0.0	0.0	0:00.83	ksoftirqd/1
23	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/1:0H-events_highpri
24	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/2
25	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/2
26	root	rt	0	0	0	0	S	0.0	0.0	0:00.70	migration/2
27	root	20	0	0	0	0	S	0.0	0.0	0:00.15	ksoftirqd/2
29	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/2:0H-events_highpri
30	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kdevtmpfs
31	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	inet_frag_wq
32	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
36	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
37	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
38	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback

ps command

```
osprac@osprac-virtual-machine:~/oslab$ ps
  PID TTY      TIME CMD
 4775 pts/0    00:00:00 bash
 9979 pts/0    00:00:00 ps
```

kill command

```
osprac@osprac-virtual-machine:~/oslab$ kill firefox
bash: kill: firefox: arguments must be process or job IDs
osprac@osprac-virtual-machine:~/oslab$ kill 7727
```

```
top - 15:19:35 up 49 min,  1 user,  load average: 0.03, 0.10, 0.16
Tasks: 299 total,  1 running, 298 sleeping,  0 stopped,  0 zombie
%Cpu(s): 1.0 us, 0.6 sy, 0.2 ni, 98.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7917.6 total, 5062.3 free, 1359.7 used, 1495.6 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used. 6195.5 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1108	osprac	20	0	5160968	340340	157572	S	3.3	4.2	2:21.39	gnome-shell
952	osprac	9	-11	1431088	27320	21656	S	1.7	0.3	0:18.60	pulseaudio
4757	osprac	20	0	596816	76956	61556	S	1.0	0.9	0:19.86	gnome-terminal
1597	osprac	20	0	2903096	81864	64028	S	0.7	1.0	0:05.28	gjs
14	root	20	0	0	0	0	I	0.3	0.0	0:06.81	rcu_sched
330	root	20	0	0	0	0	S	0.3	0.0	0:00.91	jbd2/sda3-8
540	systemd+	20	0	14824	6204	5404	S	0.3	0.1	0:04.91	systemd-oomd
1531	osprac	39	19	672796	32384	18844	S	0.3	0.4	0:01.64	tracker-miner-f
9599	root	20	0	0	0	0	I	0.3	0.0	0:00.77	kworker/u256:3-writeback
10026	osprac	20	0	908876	99504	59920	S	0.3	1.2	0:02.55	nautilus
10090	osprac	20	0	21984	4192	3228	R	0.3	0.1	0:00.01	top
1	root	20	0	166564	11744	8256	S	0.0	0.1	0:04.05	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_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_rude
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_trace
13	root	20	0	0	0	0	S	0.0	0.0	0:00.19	ksoftirqd/0
15	root	rt	0	0	0	0	S	0.0	0.0	0:00.06	migration/0
16	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
19	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/1
20	root	rt	0	0	0	0	S	0.0	0.0	0:00.70	migration/1
21	root	20	0	0	0	0	S	0.0	0.0	0:00.83	ksoftirqd/1
23	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/1:0H-events_highpri
24	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/2
25	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/2
26	root	rt	0	0	0	0	S	0.0	0.0	0:00.70	migration/2
27	root	20	0	0	0	0	S	0.0	0.0	0:00.16	ksoftirqd/2
29	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/2:0H-events_highpri
30	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kdevtmpfs
31	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	inet_frag_wq
32	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kaudit
36	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
37	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
38	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
39	root	20	0	0	0	0	S	0.0	0.0	0:00.33	kcompactd0

C) Network related commands (ping, netstat, ifconfig etc.)

```
osprac@osprac-virtual-machine:~/oslabs$ ping learning.cbit.org.in
PING learning-cbit-org-in-1413882276.ap-south-1.elb.amazonaws.com (43.204.151.98) 56(84) bytes of data.
^C
--- learning-cbit-org-in-1413882276.ap-south-1.elb.amazonaws.com ping statistics ---
35 packets transmitted, 0 received, 100% packet loss, time 34774ms

osprac@osprac-virtual-machine:~/oslabs$ ping cbit.ac.in
PING cbit.ac.in (3.111.165.12) 56(84) bytes of data.
^C
--- cbit.ac.in ping statistics ---
49 packets transmitted, 0 received, 100% packet loss, time 49120ms

osprac@osprac-virtual-machine:~/oslabs$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.5.128 netmask 255.255.255.0 broadcast 192.168.5.255
        inet6 fe80::a460:5890:b:680f prefixlen 64 scopeid 0x20<link>
          ether 00:0c:29:65:22:db txqueuelen 1000 (Ethernet)
            RX packets 87778 bytes 91429970 (91.4 MB)
            RX errors 33 dropped 33 overruns 0 frame 0
            TX packets 57601 bytes 8694021 (8.6 MB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
            device interrupt 19 base 0x2000

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 7941 bytes 1016428 (1.0 MB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 7941 bytes 1016428 (1.0 MB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
osprac@osprac-virtual-machine:~/oslabs$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0 osprac-virtual-ma:54508 ec2-52-43-253-52.:https ESTABLISHED
tcp      0      0 osprac-virtual-ma:52416 239.237.117.34.bc:https ESTABLISHED
tcp      0      0 osprac-virtual-ma:33074 20.120.65.166:https ESTABLISHED
udp      0      0 osprac-virtual-m:bootpc 192.168.5.254:bootps ESTABLISHED

Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type      State       I-Node   Path
unix    2      [ ]     DGRAM    CONNECTED  34080   /run/user/1000/systemd/notify
unix    4      [ ]     DGRAM    CONNECTED  25364   /run/systemd/notify
unix    2      [ ]     DGRAM    CONNECTED  25381   /run/systemd/journal/syslog
unix   18      [ ]     DGRAM    CONNECTED  25390   /run/systemd/journal/dev-log
unix    9      [ ]     DGRAM    CONNECTED  25392   /run/systemd/journal/socket
unix    3      [ ]     STREAM   CONNECTED  35167   /run/systemd/journal/stdout
unix    2      [ ]     DGRAM    CONNECTED  31837
unix    3      [ ]     STREAM   CONNECTED  31969
unix    3      [ ]     STREAM   CONNECTED  36190
unix    3      [ ]     STREAM   CONNECTED  36999   /run/systemd/journal/stdout
unix    2      [ ]     DGRAM    CONNECTED  121197
unix    3      [ ]     STREAM   CONNECTED  37344   /run/user/1000/bus
unix    3      [ ]     STREAM   CONNECTED  37134
unix    3      [ ]     STREAM   CONNECTED  35652   /run/user/1000/bus
unix    3      [ ]     STREAM   CONNECTED  34142   /run/dbus/system_bus_socket
unix    3      [ ]     STREAM   CONNECTED  34129   /run/systemd/journal/stdout
unix    3      [ ]     STREAM   CONNECTED  29350
unix    3      [ ]     STREAM   CONNECTED  115557  /run/systemd/journal/stdout
unix    3      [ ]     STREAM   CONNECTED  38321   /run/user/1000/bus
unix    3      [ ]     STREAM   CONNECTED  36154
unix    3      [ ]     STREAM   CONNECTED  35985   /run/user/1000/bus
unix    3      [ ]     SEQPACKET CONNECTED 135350
unix    3      [ ]     STREAM   CONNECTED  35864
unix    3      [ ]     STREAM   CONNECTED  28021   /run/systemd/journal/stdout
unix    3      [ ]     STREAM   CONNECTED  34245   /run/user/1000/bus
unix    3      [ ]     STREAM   CONNECTED  35640   /run/dbus/system_bus_socket
unix    3      [ ]     STREAM   CONNECTED  34708
unix    3      [ ]     STREAM   CONNECTED  35011   /run/dbus/system_bus_socket
unix    3      [ ]     STREAM   CONNECTED  117670  /run/user/1000/bus
unix    3      [ ]     STREAM   CONNECTED  47540   /run/dbus/system_bus_socket
unix    3      [ ]     STREAM   CONNECTED  36905   /run/user/1000/bus
unix    3      [ ]     STREAM   CONNECTED  35991   /run/user/1000/bus
unix    2      [ ]     STREAM   CONNECTED  46884
unix    2      [ ]     STREAM   CONNECTED  24757
```

2. Write a shell script to
- Check given number is prime or not
 - Check given number is palindrome number
 - Check given number is Armstrong number
 - Convert decimal to binary
 - Convert decimal to base 5
 - Convert base 5 to decimal

Ans.

- a) A program to check given number is prime or not

PROGRAM:

```
echo -n "Enter Number : "
read n
for((i=2; i<=$n/2; i++))
do
ans=$(( n%i ))
if [ $ans -eq 0 ]
then
echo "$n is not a prime number."
exit 0
fi
done
echo "$n is a prime number."
```

OUTPUT:

```
> bash main.sh
Enter Number : 100
100 is not a prime number.
>
```

- b) A program to check given number is palindrome number

PROGRAM:

```
echo "Enter Number: "
```

```
read n
num=$n
rev=0
while [ $num -gt 0 ]
do
    rev=$((rev*10 + num%10))
    num=$((num/10))
done
if [ $rev -eq $n ]
then
    echo "$n is a Palindrome"
    exit 0
fi
echo "$n is not a Palindrome"
```

OUTPUT:

```
> bash main.sh
Enter Number:
121
121 is a Palindrome
>
```

c) A program to check given number is Armstrong number

PROGRAM:

```
echo -n "Enter a number: "
read n
t=$n
sum=0
while [ $n -gt 0 ]
do
    rem=$((n%10))
    cube=$((rem*rem*rem))
    sum=$((sum+cube))
    n=$((n/10))
```

done

if [\$sum == \$t]

then

echo "\$t is a Armstrong Number"

else

echo "\$t is not a Armstrong Number"

fi

OUTPUT:

```
> bash main.sh
Enter Number:
200
200 is not a Palindrome
>
```

d) A program to convert decimal to binary

PROGRAM:

echo "Enter any decimal no:"

read num

rem=1

bno=" "

while [\$num -gt 0]

do

rem=`expr \$num % 2 `

bno=\$bno\$rem

num=`expr \$num / 2 `

done

i=\${#bno}

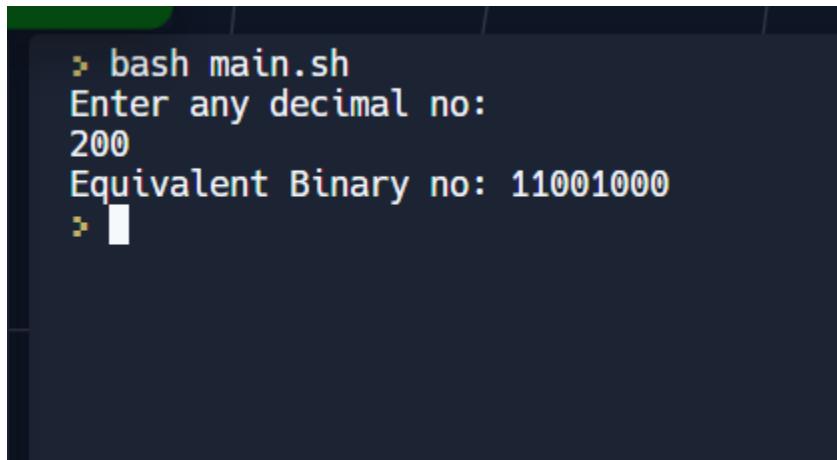
final=" "

while [\$i -gt 0]

do

rev=`echo \$bno | awk '{ printf substr(\$0,'\$i',1) }`

```
final=$final$rev  
i=$(( $i - 1 ))  
done  
echo "Equivalent Binary no:" $final
```

OUTPUT:

```
> bash main.sh  
Enter any decimal no:  
200  
Equivalent Binary no: 11001000  
>
```

- e) A program to convert decimal to base 5

PROGRAM:

```
echo -n "Enter any decimal no: "  
read num  
rem=1  
bno=""  
while [ $num -gt 0 ]  
do  
rem=$((num%5))  
bno=$rem$bno  
num=$((num/5))  
done  
echo "The converted base 5 number: $bno"
```

OUTPUT:

```
> bash main.sh
Enter any decimal no: 200
The converted base 5 number: 1300
> █
```

f) A program to convert base 5 to decimal**PROGRAM:**

```
echo -n "Enter Base 5 number: "
```

```
read bno
```

```
dec=0
```

```
pow=1
```

```
while [ $bno -gt 0 ]
```

```
do
```

```
val=$((bno%10))
```

```
dec=$((dec+val*pow))
```

```
bno=$((bno/10))
```

```
pow=$((pow*5))
```

```
done
```

```
echo "The Decimal number is: $dec"
```

OUTPUT:

```
> bash main.sh
Enter Base 5 number: 1300
The Decimal number is: 200
> █
```

OS LAB WEEK – 3

1. Demonstrate System calls with simple example programs

Ans.

PROGRAM:

```
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>

int main( ){
    pid_t child_pid;
    child_pid = fork (); // Create a new child process;
    if (child_pid < 0) {
        printf("fork failed");
        return 1;
    } else if (child_pid == 0) {
        printf ("child process successfully created!\n");
        printf ("child_PID = %d,parent_PID = %d\n",
               getpid(), getppid( ) );
    } else {
        // wait(NULL);
        printf ("parent process successfully created!\n");
        printf ("child_PID = %d, parent_PID = %d", getpid( ), getppid( ) );
    }
    return 0;
}
```

OUTPUT:

```
ubuntu@ubuntu:~/Desktop/oslab$ gcc programs.c
ubuntu@ubuntu:~/Desktop/oslab$ ./a.out
parent process successfully created!
child_PID = 10970, parent_PID = 6958child process successfully created!
ubuntu@ubuntu:~/Desktop/oslab$ child_PID = 10971, parent_PID = 2436

ubuntu@ubuntu:~/Desktop/oslab$ █
```

2. Demonstrate following I/O system calls with examples

- i. Open
- ii. Read
- iii. Write
- iv. Lseek
- v. ioctl
- vi. Stat
- vii. Syn

Ans.

PROGRAM:

```
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<stdlib.h>
#include<sys/ioctl.h>
#include<sys/stat.h>

#define WR_VALUE _IOW('a','a',int*)
#define RD_VALUE _IOR('a','b',int*)

int main()
{
    // if file does not have in directory
    // then file foo.txt is created.

    int fd = open("sample.txt", O_RDONLY | O_CREAT);
    int fd2 = open("file2.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
    int fd3 = open("file3.txt", O_RDWR);
    char c[20];
    // reading from sample.txt
    read(fd, &c, 20);
    printf("Text in file is: %s\n",c);
    //writing into file2.txt which was stored in c
    write(fd2, c, read(fd, &c, 20));
    lseek(fd, 5, SEEK_CUR);
    char c1[20];
```

```
    read(fd, &c1, 20);

    printf("Text after lseek 10 characters: %s",c1);

    printf("Enter a value to write in the file: ");

    int number;

    scanf("%d",&number);

    ioctl(fd3, WR_VALUE,(int*) &number);

    struct stat sfile;

    stat("stat.c", &sfile);

    printf("st_mode = %o\n",sfile.st_mode);

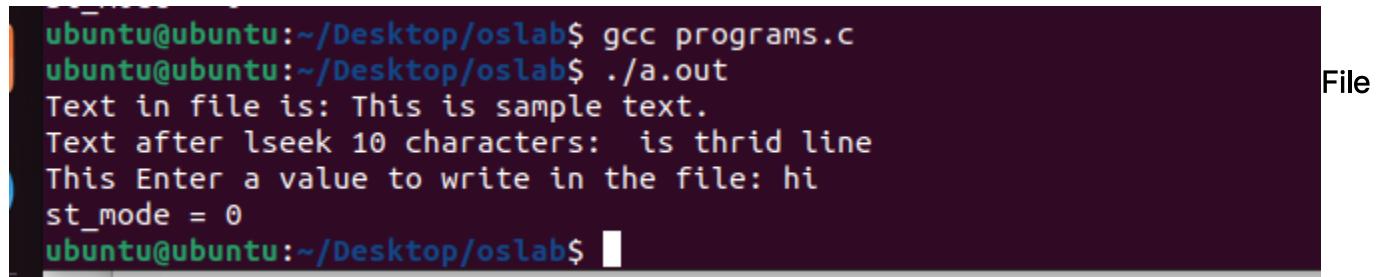
    close(fd);

    close(fd2);

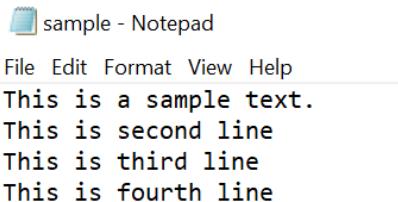
    close(fd3);

    return 0;

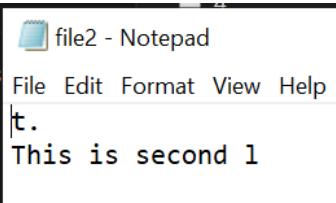
}
```

OUTPUT:**Terminal output:**

```
ubuntu@ubuntu:~/Desktop/oslab$ gcc programs.c
ubuntu@ubuntu:~/Desktop/oslab$ ./a.out
Text in file is: This is sample text.
Text after lseek 10 characters: is thrid line
This Enter a value to write in the file: hi
st_mode = 0
ubuntu@ubuntu:~/Desktop/oslab$
```

output:**i. Sample.txt**

```
sample - Notepad
File Edit Format View Help
This is a sample text.
This is second line
This is third line
This is fourth line
```

ii. File2.txt

```
file2 - Notepad
File Edit Format View Help
t.
This is second 1
```

3. Demonstrate following file system call with examples

i. Link

```
ubuntu@ubuntu:~/Desktop/oslab$ cat > test.txt
This is test file which is about to be linked to other file
^X
^C
ubuntu@ubuntu:~/Desktop/oslab$ link test.txt linked.txt
ubuntu@ubuntu:~/Desktop/oslab$ cat linked.txt
This is test file which is about to be linked to other file
```

ii. Unlink

```
ubuntu@ubuntu:~/Desktop/oslab$ ls
a.out file2.txt file3.txt linked.txt programs.c sample.txt test.txt
ubuntu@ubuntu:~/Desktop/oslab$ unlink linked.txt
ubuntu@ubuntu:~/Desktop/oslab$ ls
a.out file2.txt file3.txt programs.c sample.txt test.txt
ubuntu@ubuntu:~/Desktop/oslab$
```

iii. Mount

```
ubuntu@ubuntu:~/Desktop/oslab$ mount -V
mount from util-linux 2.37.2 (libmount 2.37.2: selinux, smack, btrfs, verity, namespaces, assert, debug)
ubuntu@ubuntu:~/Desktop/oslab$
```

iv. Umount

```
ubuntu@ubuntu:~/Desktop/oslab$ umount -V
umount from util-linux 2.37.2 (libmount 2.37.2: selinux, smack, btrfs, verity, namespaces, assert, debug)
ubuntu@ubuntu:~/Desktop/oslab$
```

v. Chmod

```
ubuntu@ubuntu:~/Desktop/oslab$ ls
a.out file2.txt file3.txt programs.c sample.txt test.txt
ubuntu@ubuntu:~/Desktop/oslab$ ls -lrt
total 32
-rw-rw-r-- 1 ubuntu ubuntu 1054 Sep 27 23:36 programs.c
-rw-rw-r-- 1 ubuntu ubuntu   80 Sep 27 23:40 sample.txt
-rw-rw-r-- 1 ubuntu ubuntu     0 Sep 27 23:40 file3.txt
-rwxrwxr-x 1 ubuntu ubuntu 16344 Sep 27 23:40 a.out
-rw-rw-r-- 1 ubuntu ubuntu    20 Sep 27 23:40 file2.txt
-rw-rw-r-- 1 ubuntu ubuntu    62 Sep 27 23:43 test.txt
ubuntu@ubuntu:~/Desktop/oslab$ chmod 777 test.txt
ubuntu@ubuntu:~/Desktop/oslab$ ls -lrt
total 32
-rw-rw-r-- 1 ubuntu ubuntu 1054 Sep 27 23:36 programs.c
-rw-rw-r-- 1 ubuntu ubuntu   80 Sep 27 23:40 sample.txt
-rw-rw-r-- 1 ubuntu ubuntu     0 Sep 27 23:40 file3.txt
-rwxrwxr-x 1 ubuntu ubuntu 16344 Sep 27 23:40 a.out
-rw-rw-r-- 1 ubuntu ubuntu    20 Sep 27 23:40 file2.txt
-rwxrwxrwx 1 ubuntu ubuntu    62 Sep 27 23:43 test.txt
ubuntu@ubuntu:~/Desktop/oslab$
```

```
ubuntu@ubuntu:~/Desktop/oslab$ sudo chown -c root test.txt
[sudo] password for ubuntu:
ubuntu is not in the sudoers file. This incident will be reported.
ubuntu@ubuntu:~/Desktop/oslab$ 
```

4. Demonstrate how process is created and allocate memory in /proc folder with example program

Ans.

PROGRAM:

```
#include<stdio.h>
#include<fcntl.h>
#include<unistd.h>

int main(int argc, char *argv[])
{
    int a;

    int fd1=open("kk.txt",O_RDONLY,742); //open system call
    int fd2=open("kk1.txt",O_RDONLY,S_IRWXU|S_IRGRP|S_IXOTH);
    printf(" fd1=%d,fd2=%d\n", fd1,fd2);
    printf("Process Id=%d\n",getpid());
    scanf("%d",&a);
}
```

OUTPUT:

Process program execution output:

```
ubuntu@ubuntu:~/Desktop/oslab$ gcc programs.c
ubuntu@ubuntu:~/Desktop/oslab$ ./a.out
fd1=-1,fd2=-1
Process Id=11563
```

in proc folder

```
ubuntu@ubuntu:~/proc$ ls
1 1040 111 12108 1282 1374 1462 1550 1716 24 2500 2748 2897 2950 3012 3113 37 659 742 94  devices  keys  net  timer_list
10 105 112 12109 1283 1386 1463 1580 1762 2422 2514 2753 2909 2955 3019 312 374 667 745 97  diskstats  key-users  pagetypeinfo  tty
100 10592 12177 129 14 1465 1581 1765 2436 252 2758 2911 2972 302 3135 38 670 769 98  dma  ksm  partitions  uptime
1000 106 113 12221 1295 1414 1468 1584 1768 2437 2564 276 2921 2973 3020 3162 39 672 86 986  driver  kpagegroup  pressure  version
1008 1063 12231 1295 1414 1468 1584 1768 2437 2564 276 2921 2973 3020 3162 39 672 86 986  dynamic_debug  kpagecount  schedstat  version_signature
1010 1063 12231 1295 1414 1468 1584 1768 2437 2564 276 2921 2973 3020 3162 39 672 86 986  kpageflags  self  vmallocinfo
1011 1063 12423 1300 1432 1470 1590 18 2444 26 28 2924 299 3035 42 482 77 873  acpi  execdomains  loadavg  self  vnstat
1013 10683 12355 1308 1438 1434 1477 16 1832 2445 2605 2838 2925 2996 3036 3209 484 682 88  asound  fb  modules  sys
1015 10698 11481 12468 1321 1437 1494 1640 1833 2454 2625 2842 2927 2999 304 3262 5 684 884  bootconfig  filesystems  locks  slabinfo  zoneinfo
1017 10700 12471 1333 1441 1470 1590 19 2444 26 28 2924 299 3035 34 604 6859 886  buddyinfo  mdstat  softirqs
1020 10897 1157 12441 1333 1441 1470 1590 19 2444 26 28 2924 299 3035 34 604 6859 886  bus  interrupts  meminfo  stat
1021 109 1376 1245 134 1454 1518 1635 20 2472 2649 286 2935 300 3057 345 652 6860 9  cgroups  ionem  misc  swaps
1026 10912 1180 125 1343 1454 1524 1658 21 2481 2677 2866 2936 3002 306 35 653 688 90  cmdline  loports  modules  sys
1030 11 1188 1251 1352 1455 1526 1672 211 2488 2688 2874 2942 3008 307 3523 655 693 91  consoles  irq  mounts  sysrq-trigger
1032 11010 128 1354 1456 1526 1678 212 2491 27 2880 2966 3009 31 36 656 696 92  cpulinfo  kallsyms  mtd  sysvIPC
104 1104 12002 128 1354 1460 1549 1712 22 25 2726 2894 2946 301 3101 369 657 7 93  crypto  kcore  mtrr  thread-self
104 1104 12002 128 1374 1462 1550 1716 24 2500 2748 2897 2950 3012 3113 37 659 742 94  devices  keys  net  timer_list
ubuntu@ubuntu:~/proc$ ls
```

OS LAB WEEK – 3

1. Write a program to demonstrate the use of fork() system call

Ans.

PROGRAM :

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

int main()
{
    printf(" Before fork");

    int fd1=open("kk.txt",O_RDONLY,742); //kk.txt is a shared resource to parent and child process

    int a; // a is shared variable to parent and child process

    printf(" Process Id of parent %d \n",getpid());

    int rf=fork();

    if(rf==0) // Upon successful execution fork() --> Child process will execute below code in if block

    {
        printf("Inside child process \n");

        printf(" Process Id of child pid= %d rf=%d \n ",getpid(),rf);

        scanf("%d",&a);

        printf(" Inside child a value is a=%d",a);

    }

    else if (rf> 0) // Upon successful execution fork() system call --> Parent process will execute below code in elseif block

    {
        printf("Inside parent process");

        printf(" Process Id of parent pid=%d rf=%d \n ",getpid(),rf);

        scanf("%d",&a);

        printf(" Inside parent a value is a=%d",a);

    }

    else{ // Upon unsuccessful execution fork() system call Parent process will execute below code in else block
}
}
```

```

printf(" Fork unsucessful \n Indide parent and Process Id of parent pid= %d rf=%d \n ",getpid(),rf);
}

printf("End of process %d", getpid());

return 0;

}

```

OUTPUT:

```

Lab/week 3$ ./1
Before fork Process Id of parent 96
Inside parent process Process Id of parent pid=96 rf=97
Inside child process
Process Id of child pid= 97 rf=0
12
Inside child a value is a=12End of process 97
1

```

In proc folder:

```

gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ ls
1  3_4  5          exec      execv.c    forkex2.c  orphan.c      vfork.c
2  3_5  6          exec.c    execvp.c   kk1.txt    prime_factorial.c zombie.c
3_1 4  demoprocess.c  execl.c   forkex1.c  kk.txt    sree
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ █

```

2. Write a program to demonstrate the use of exe system calls

Parent process → Execute factorial of given number

Child process → Execute prime number

Ans.

PROGRAM :

```

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

```

```

int main()
{
    printf(" Before fork");

    int fd1=open("kk.txt",O_RDONLY,742); //kk.txt is a shared resource to parent and child process

    int a; // a is shared variable to parent and child process

    printf(" Process Id of parent %d \n",getpid());

    int rf=fork();

    if(rf==0) // Upon successful execution fork() --> Child process will execute below code in if block

```

```

{
    printf("Inside child process \n");
    printf(" Process Id of child pid= %d rf=%d \n ",getpid(),rf);
    printf("Child process: to execute the prime number program\nEnter a number: ");
    scanf("%d",&a);

    int flag = 0;
    for(int i=2;i<=a/2;i++){
        if (a%i==0){
            flag=1;
            break;
        }
    }
    if(flag==0){
        printf("Given number %d is a prime number\n",a);
    }
    else{
        printf("Given number %d is not a prime number\n",a);
    }
}

else if (rf> 0) // Upon successful execution fork() system call --> Parent process will execute below code in elseif
block
{
    printf("Inside parent process");
    printf(" Process Id of parent pid=%d rf=%d \n",getpid(),rf);
    printf("Parent process: to execute the factorial of a number\nEnter a number: ");
    scanf("%d",&a);
    int total = 1, i = 1;
    while(i<=a){
        total=total*i;
        i=i+1;
    }
    printf("Factorial value is: %d\n",total);
}

```

```

else{ // Upon uncessful execution fork() system call Parent process will exceute below code in else block
    printf(" Fork unsuccesful \n Indide parent and Process Id of parent pid= %d rf=%d \n ",getpid(),rf);
}

printf("End of process %d\n", getpid());

return 0;
}

```

OUTPUT:

```

gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315$ cd week\ 3/
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ gcc prime_factorial.c -o 1
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ ./1
Before fork Process Id of parent 4386
Inside parent process Process Id of parent pid=4386 rf=4387
Parent process: to execute the factorial of a number
Enter a number: Inside child process
Process Id of child pid= 4387 rf=0
Child process: to execute the prime number program
Enter a number: 5
Factorial value is: 120
End of process 4386
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ Given number 5 is
a prime number
End of process 4387

```

3. Write a program to demonstrate use following exe functions

- int execl(const char* path, const char* arg, ...)
- int execlp(const char* file, const char* arg, ...)
- int execle(const char* path, const char* arg, ..., char* const envp[])
- int execv(const char* path, const char* argv[])
- int execvp(const char* file, const char* argv[])
- int execvpe(const char* file, const char* argv[], char *const envp[])

Program:**Exec.c**

```

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

int main()
{
    int i;
    printf("I am Demo file called by execvp() ");
    printf("\n");
}

```

Execv.c

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main()
{
    char *args[]={ "./exec",NULL};
    execv(args[0],args);
    printf("Ending----");
    return 0;
}
```

Output:

```
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ ./3_4
I am Demo file called by family of exec() functions
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$
```

Execvp.c

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main()
{
    char *args[]={ "./3_6_demo",NULL};
    execvp(args[0],args
    printf("Ending----");
    return 0;
}
```

Output:

```
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ gcc exec.c -o exec
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ gcc execvp.c -o 3_5
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ ./3_5
I am Demo file called by family of exec() functions
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$
```

4. Write a program to demonstrate the use of vfork()

Ans.

PROGRAM :

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

int main()
{
    printf(" Before fork");

    int fd1=open("kk.txt",O_RDONLY,742); //kk.txt is a shared resource to parent and child process

    int a,b; // a is shared variable to parent and child process

    printf(" Process Id of parent %d \n",getpid());

    int rf=vfork();

    if(rf==0) // Upon successful execution fork() --> Child process will execute below code in if block
    {
        printf("Inside child process \n");

        printf(" Process Id of child pid= %d rf=%d \n ",getpid(),rf);
        scanf("%d",&b);

        printf(" Inside child b value is b=%d",b);

    }

    else if (rf> 0) // Upon successful execution fork() system call --> Parent process will execute below code in elseif
block
    {
```

```

        printf("Inside parent process");

        printf(" Process Id of parent pid=%d rf=%d \n",getpid(),rf);

        scanf("%d",&a);

        printf(" Inside parent a value is a=%d",a);

    }

else{ // Upon uncessful execution fork() system call Parent process will execute below code in else block

    printf(" Fork unsucessful \n Indide parent and Process Id of parent pid= %d rf=%d \n ",getpid(),rf);

}

printf("End of process %d\n", getpid());

return 0;

}

```

OUTPUT:

```

I am demostrate called by rarely or exec() functions
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ gcc vfork.c -o 4
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ ./4
Before fork Process Id of parent 4592
Inside child process
Process Id of child pid= 4593  rf=0
9
Inside child b value is b=9End of process 4593
Inside parent process Process Id of parent pid=4592 rf=4593
2
Inside parent a value is a=2End of process 4592
*** stack smashing detected ***: terminated
Aborted (core dumped)
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ █

```

5. Write a program to demonstrate Orphan process**Ans.****PROGRAM :**

```

#include<stdio.h>

#include <sys/types.h>
#include <unistd.h>

// A C program to demonstrate Orphan Process.

// Parent process finishes execution while the
// child process is running. The child process
// becomes orphan.

int main()

```

```
{  
    // Create a child process  
    int pid = fork();  
  
    if (pid > 0)  
        printf("in parent process\n");  
  
    // Note that pid is 0 in child process  
    // and negative if fork() fails  
    else if (pid == 0)  
    {  
        sleep(9);  
        printf("in child process\n");  
    }  
}
```

OUTPUT:

```
ABORTED (core dumped)  
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ gcc orphan.c -o 5  
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ ./5  
in parent process  
gadhamsrinivas@gadhamsrinivas-VirtualBox:~/Desktop/315/week 3$ in child processin child process
```

6. Write a program to demonstrate Zombie process**Ans.****PROGRAM :**

```
#include <stdlib.h>  
  
#include <sys/types.h>  
  
#include <sys/wait.h>  
  
#include <unistd.h>  
  
#include <stdio.h>  
  
int main()  
{  
    pid_t pid;  
    char *message;  
    int n;  
    int exit_code;
```

```
printf("fork program starting\n");
```

```
pid = fork();
```

```
switch(pid)
```

```
{
```

```
case -1:
```

```
    perror("fork failed");
```

```
    exit(1);
```

```
case 0:
```

```
    message = "This is the child";
```

```
    n = 3;
```

```
    exit_code = 37;
```

```
break;
```

```
default:
```

```
    message = "This is the parent";
```

```
    n = 5;
```

```
    exit_code = 0;
```

```
break;
```

```
}
```

```
for(; n > 0; n--) {
```

```
    puts(message);
```

```
    sleep(1);
```

```
}
```

```
}
```

OUTPUT:

```
gadham.srinivas@gadham.srinivas-VirtualBox:~/Desktop/315/week 3$ gcc forkex2.c -o 6
gadham.srinivas@gadham.srinivas-VirtualBox:~/Desktop/315/week 3$ ./6
fork program starting
This is the parent
This is the child
This is the parent
This is the child
This is the parent
This is the child
This is the parent
This is the parent
gadham.srinivas@gadham.srinivas-VirtualBox:~/Desktop/315/week 3$ █
```

WEEK-5

1)

Aim: Write a program to demonstrate First Come First Serve CPU scheduling

Program:

```
#include<stdio.h>
int main()
{
    int p[10],at[10],bt[10],ct[10],tat[10],wt[10],i,j,temp=0,n;
    float awt=0,atat=0;
    printf("enter no of process you want:");
    scanf("%d",&n);
    printf("enter %d process:",n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&p[i]);
    }
    printf("enter %d arrival time:",n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&at[i]);
    }
    printf("enter %d burst time:",n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&bt[i]);
    }
    for(i=0;i<n;i++)
    {
        for(j=0;j<(n-i);j++)
        {
            if(at[j]>at[j+1])
            {
                temp=p[j+1];
                p[j+1]=p[j];
                p[j]=temp;
                temp=at[j+1];
                at[j+1]=at[j];
                at[j]=temp;
                temp=bt[j+1];
                bt[j+1]=bt[j];
                bt[j]=temp;
            }
        }
    }
    ct[0]=at[0]+bt[0];
    for(i=1;i<n;i++)
    {
        temp=0;
        if(ct[i-1]<at[i])
```

```

{
    temp=at[i]-ct[i-1];
}
ct[i]=ct[i-1]+bt[i]+temp;
printf("\n\nP\t A.T\t B.T\t C.T\t TAT\t WT");
for(i=0;i<n;i++)
{
    tat[i]=ct[i]-at[i];
    wt[i]=tat[i]-bt[i];
    atat+=tat[i];
    awt+=wt[i];
}
atat=atat/n;
awt=awt/n;
for(i=0;i<n;i++)
{
    printf("\nP%d\t %d\t %d\t %d\t %d\t %d",p[i],at[i],bt[i],ct[i],tat[i],wt[i]);
}
printf("\naverage turnaround time is %f",atat);

printf("\naverage waiting timme is %f\n",awt);
return 0;
}

```

Output:

```

enter no of process you want:4
enter 4 process:1 2 3 4
enter 4 arrival time:3 4 1 2
enter 4 burst time:1 2 2 1

      P      A.T      B.T      C.T      TAT      WT
P8      0      0      0      0      0
P3      1      2      3      2      0
P4      2      1      4      2      1
P1      3      1      5      2      1
average turnaround time is 1.500000
average waiting timme is 0.500000

```

2)

Aim: Write a program to demonstrate Shortest Job First or Shortest Job next CPU scheduling

Program:(preemptive)

```

#include <stdio.h>
int main()
{
    int arrival_time[10], burst_time[10], temp[10];
    int i, smallest, count = 0, time, limit;
    double wait_time = 0, turnaround_time = 0, end;
    float average_waiting_time, average_turnaround_time;
    printf("\nEnter the Total Number of Processes:\t");

```

```
scanf("%d", &limit);
printf("\nEnter Details of %d Processes\n", limit);
for(i = 0; i < limit; i++)
{
    printf("\nEnter Arrival Time:\t");
    scanf("%d", &arrival_time[i]);
    printf("Enter Burst Time:\t");
    scanf("%d", &burst_time[i]);
    temp[i] = burst_time[i];
}
burst_time[9] = 9999;
for(time = 0; count != limit; time++)
{
    smallest = 9;
    for(i = 0; i < limit; i++)
    {
        if(arrival_time[i] <= time && burst_time[i] < burst_time[smallest] &&
burst_time[i] > 0)
        {
            smallest = i;
        }
    }
    burst_time[smallest]--;
    if(burst_time[smallest] == 0)
    {
        count++;
        end = time + 1;
        wait_time = wait_time + end - arrival_time[smallest] - temp[smallest];
        turnaround_time = turnaround_time + end - arrival_time[smallest];
    }
}
average_waiting_time = wait_time / limit;
average_turnaround_time = turnaround_time / limit;
printf("\n\nAverage Waiting Time:\t%lf\n", average_waiting_time);
printf("Average Turnaround Time:%lf\n", average_turnaround_time);
return 0;
}
```

Output:

```
Enter the Total Number of Processes:      3
Enter Details of 3 Processes
Enter Arrival Time:          1
Enter Burst Time:           4
Enter Arrival Time:          2
Enter Burst Time:           6
Enter Arrival Time:          3
Enter Burst Time:           3

Average Waiting Time:   2.666667
Average Turnaround Time:7.000000
```

Program(non preemptive):

```
#include<stdio.h>
int main()
{
    int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
    float avg_wt,avg_tat;
    printf("Enter number of process:");
    scanf("%d",&n);

    printf("\nEnter Burst Time:");
    for(i=0;i<n;i++)
    {
        printf("p%d:",i+1);
        scanf("%d",&bt[i]);
        p[i]=i+1;
    }

    //sorting of burst times
    for(i=0;i<n;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
            if(bt[j]<bt[pos])
                pos=j;
        }

        temp=bt[i];
        bt[i]=bt[pos];
        bt[pos]=temp;

        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
    }

    wt[0]=0;

    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];

        total+=wt[i];
    }

    avg_wt=(float)total/n;
    total=0;
```

```

printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{
    tat[i]=bt[i]+wt[i];
    total+=tat[i];
    printf("\np%d\t %d\t %d",p[i],bt[i],tat[i]);
}

avg_tat=(float)total/n;
printf("\n\nAverage Waiting Time=%f",avg_wt);
printf("\nAverage Turnaround Time=%f\n",avg_tat);
}

```

Output:

```

Enter number of process:4

Enter Burst Time:np1:3
p2:3
p3:5
p4:8

Process      Burst Time          Waiting Time       Turnaround Time
p1            3                  0                  3
p2            3                  3                  6
p3            5                  6                  11
p4            8                  11                 19

Average Waiting Time=5.000000
Average Turnaround Time=9.750000

```

3)**Aim:** Write a program to demonstrate Round Robin CPU scheduling**Program:**

```

#include<stdio.h>
int main()
{
    int i, limit, total = 0, x, counter = 0, time_quantum;
    int wait_time = 0, turnaround_time = 0, arrival_time[10], burst_time[10], temp[10];
    float average_wait_time, average_turnaround_time;
    printf("\nEnter Total Number of Processes:\t");
    scanf("%d", &limit);
    x = limit;
    for(i = 0; i < limit; i++)
    {
        printf("\nEnter Details of Process[%d]\n", i + 1);

        printf("Arrival Time:\t");

        scanf("%d", &arrival_time[i]);

        printf("Burst Time:\t");

        scanf("%d", &burst_time[i]);

        temp[i] = burst_time[i];
    }
}

```

```
printf("\nEnter Time Quantum:\t");
scanf("%d", &time_quantum);
printf("\nProcess ID\tBurst Time\t Turnaround Time\t Waiting Time\n");
for(total = 0, i = 0; x != 0;
{
    if(temp[i] <= time_quantum && temp[i] > 0)
    {
        total = total + temp[i];
        temp[i] = 0;
        counter = 1;
    }
    else if(temp[i] > 0)
    {
        temp[i] = temp[i] - time_quantum;
        total = total + time_quantum;
    }
    if(temp[i] == 0 && counter == 1)
    {
        x--;
        printf("\nProcess[%d]\t%d\t %d\t %d\t %d", i + 1, burst_time[i], total - arrival_time[i], total - arrival_time[i] - burst_time[i]);
        wait_time = wait_time + total - arrival_time[i] - burst_time[i];
        turnaround_time = turnaround_time + total - arrival_time[i];
        counter = 0;
    }
    if(i == limit - 1)
    {
        i = 0;
    }
    else if(arrival_time[i + 1] <= total)
    {
        i++;
    }
    else
    {
        i = 0;
    }
}
average_wait_time = wait_time * 1.0 / limit;
average_turnaround_time = turnaround_time * 1.0 / limit;
printf("\n\nAverage Waiting Time:\t%.2f", average_wait_time);
printf("\nAvg Turnaround Time:\t%.2f\n", average_turnaround_time);
return 0;
}
```

Output:

```

Enter Details of Process[2]
Arrival Time: 1
Burst Time: 4

Enter Details of Process[3]
Arrival Time: 3
Burst Time: 6

Enter Details of Process[4]
Arrival Time: 1
Burst Time: 5

Enter Time Quantum: 2

Process ID           Burst Time       Turnaround Time      Waiting Time
Process[2]            4                  11                   7
Process[3]            6                  17                   11
Process[4]            5                  20                   15
Process[1]            7                  20                   13

Average Waiting Time: 11.500000
Avg Turnaround Time: 17.000000

```

4)

Aim: Write a program to demonstrate Priority CPU Scheduling

Program(preemptive):

```

#include<stdio.h>
struct process
{
    int WT,AT,BT,TAT,PT;
};

struct process a[10];

int main()
{
    int n,temp[10],t,count=0,short_p;
    float total_WT=0,total_TAT=0,Avg_WT,Avg_TAT;
    printf("Enter the number of the process\n");
    scanf("%d",&n);
    printf("Enter the arrival time , burst time and priority of the process\n");
    printf("AT BT PT\n");
    for(int i=0;i<n;i++)
    {
        scanf("%d%d%d",&a[i].AT,&a[i].BT,&a[i].PT);

        // copying the burst time in
        // a temp array for further use
        temp[i]=a[i].BT;
    }

    // we initialize the burst time
    // of a process with maximum
    a[9].PT=10000;

    for(t=0;count!=n;t++)
    {

```

```
short_p=9;
for(int i=0;i<n;i++)
{
    if(a[short_p].PT>a[i].PT && a[i].AT<=t && a[i].BT>0)
    {
        short_p=i;
    }
}

a[short_p].BT=a[short_p].BT-1;

// if any process is completed
if(a[short_p].BT==0)
{
    // one process is completed
    // so count increases by 1
    count++;
    a[short_p].WT=t+1-a[short_p].AT-temp[short_p];
    a[short_p].TAT=t+1-a[short_p].AT;

    // total calculation
    total_WT=total_WT+a[short_p].WT;
    total_TAT=total_TAT+a[short_p].TAT;
}

}

Avg_WT=total_WT/n;
Avg_TAT=total_TAT/n;

// printing of the answer
printf("ID WT TAT\n");
for(int i=0;i<n;i++)
{
    printf("%d %d\t%d\n",i+1,a[i].WT,a[i].TAT);
}

printf("Avg waiting time of the process is %f\n",Avg_WT);
printf("Avg turn around time of the process is %f\n",Avg_TAT);

return 0;
}
```

Output:

```

Enter the number of the process
5
Enter the arrival time , burst time and priority of the process
AT BT PT
1 4 1
2 6 2
1 4 3
2 5 1
3 7 2
ID WT TAT
1 0      4
2 8      14
3 22     26
4 3      8
5 13     20
Avg waiting time of the process is 9.200000
Avg turn around time of the process is 14.400000

```

Program(Non preemptive):

```

#include<stdio.h>
struct process
{
    int id,WT,AT,BT,TAT,PR;
};
struct process a[10];

// function for swapping
void swap(int *b,int *c)
{
    int tem;
    tem=*c;
    *c=*b;
    *b=tem;
}

//Driver function
int main()
{
    int n,check_ar=0;
    int Cmp_time=0;
    float Total_WT=0,Total_TAT=0,Avg_WT,Avg_TAT;
    printf("Enter the number of process \n");
    scanf("%d",&n);
    printf("Enter the Arrival time , Burst time and priority of the process\n");
    printf("AT BT PR\n");
    for(int i=0;i<n;i++)
    {
        scanf("%d%d%d",&a[i].AT,&a[i].BT,&a[i].PR);
        a[i].id=i+1;
        // here we are checking that arrival time
        // of the process are same or different
        if(i==0)
            check_ar=a[i].AT;

```

```

if(check_ar!=a[i].AT )
check_ar=1;

}

// if process are arrived at the different time
// then sort the process on the basis of AT
if(check_ar!=0)
{
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n-i-1;j++)
        {
            if(a[j].AT>a[j+1].AT)
            {
                swap(&a[j].id,&a[j+1].id);
                swap(&a[j].AT,&a[j+1].AT);
                swap(&a[j].BT,&a[j+1].BT);
                swap(&a[j].PR,&a[j+1].PR);
            }
        }
    }
}

// logic of Priority scheduling ( non preemptive) algo
// if all the process are arrived at different time
if(check_ar!=0)
{
    a[0].WT=a[0].AT;
    a[0].TAT=a[0].BT-a[0].AT;
    // cmp_time for completion time
    Cmp_time=a[0].TAT;
    Total_WT=Total_WT+a[0].WT;
    Total_TAT=Total_TAT+a[0].TAT;
    for(int i=1;i<n;i++)
    {
        int min=a[i].PR;
        for(int j=i+1;j<n;j++)
        {
            if(min>a[j].PR && a[j].AT<=Cmp_time)
            {
                min=a[j].PR;
                swap(&a[i].id,&a[j].id);
                swap(&a[i].AT,&a[j].AT);
                swap(&a[i].BT,&a[j].BT);
                swap(&a[i].PR,&a[j].PR);
            }
        }
        a[i].WT=Cmp_time-a[i].AT;
        Total_WT=Total_WT+a[i].WT;
    }
}

```

```
// completion time of the process
Cmp_time=Cmp_time+a[i].BT;

// Turn Around Time of the process
// compl-Arrival
a[i].TAT=Cmp_time-a[i].AT;
Total_TAT=Total_TAT+a[i].TAT;

}

}

// if all the process are arrived at same time
else
{
    for(int i=0;i<n;i++)
    {
        int min=a[i].PR;
        for(int j=i+1;j<n;j++)
        {
            if(min>a[j].PR && a[j].AT<=Cmp_time)
            {
                min=a[j].PR;
                swap(&a[i].id,&a[j].id);
                swap(&a[i].AT,&a[j].AT);
                swap(&a[i].BT,&a[j].BT);
                swap(&a[i].PR,&a[j].PR);
            }
        }
        a[i].WT=Cmp_time-a[i].AT;

        // completion time of the process
        Cmp_time=Cmp_time+a[i].BT;

        // Turn Around Time of the process
        // compl-Arrival
        a[i].TAT=Cmp_time-a[i].AT;
        Total_WT=Total_WT+a[i].WT;
        Total_TAT=Total_TAT+a[i].TAT;
    }
}

Avg_WT=Total_WT/n;
Avg_TAT=Total_TAT/n;

// Printing of the results
printf("The process are\n");
printf("ID WT TAT\n");
for(int i=0;i<n;i++)
{
```

```
    printf("%d\t%d\t%d\n",a[i].id,a[i].WT,a[i].TAT);
}

printf("Avg waiting time is: %f\n",Avg_WT);
printf("Avg turn around time is: %f",Avg_TAT);
return 0;
}
```

Output:

```
Enter the number of process
4
Enter the Arrival time , Burst time and priority of the process
AT BT PR
1 3 5
2 6 2
3 8 1
1 2 1
The process are
ID WT TAT
1      1      2
4      1      3
3      1      9
2     10     16
Avg waiting time is: 3.250000
Avg turn around time is: 7.500000 ]
```

OS Lab Week – 6

1. Write a program to demonstrate the use of atexit() exit handler

Ans.

PROGRAM:

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

//File Descriptors
int fd1;
int fd2;

void functionA () { // Exit Handler
    close(fd1);
    printf("We are in functionA and closed the file fd1 \n");
}

void functionB () { // Exit Handler
    close(fd2);
    printf("We are in functionB and closed the file fd2\n");
}

int main () {
    /* register the termination function */
    char buf[10];
    fd1=open("file1.txt",O_RDONLY,742);
    fd2=open("file2.txt",O_RDONLY,S_IRWXU|S_IRGRP|S_IOTH);
    read(fd1,buf,10);
    write(1,buf,10);
    printf("\nCompleted the operation A \n");
    atexit(functionA );
    read(fd2,buf,10);
    write(1,buf,10);
    printf("\nCompleted the operation B \n");
    atexit(functionB );
}
```

```

printf("Starting main program...\n");
printf("Exiting main program...\n");
return(0);
}

```

OUTPUT

```

we are in functionA and closed the file fd1
gadhamsrinivas@ESKTOP-P2DEFRB:~/oslab$ gcc exithandler.c -o 1
gadhamsrinivas@ESKTOP-P2DEFRB:~/oslab$ ./1
hello sree
Completed the operation A
This is me
Completed the operation B
Starting main program...
Exiting main program...
We are in functionB and closed the file fd2
We are in functionA and closed the file fd1
gadhamsrinivas@ESKTOP-P2DEFRB:~/oslab$ -

```

2. Write a program to demonstrate the use of exit() and _exit() system call**Ans.****PROGRAM:****exit() function:**

```

#include<stdio.h>
#include<stdlib.h>
void func (){
    printf("Exiting from function\n");
}
int main()
{
    atexit(func);
    printf("Exiting from main()\n");
    exit(0);
}

```

_Exit() function:

```

#include<stdio.h>
#include<stdlib.h>
void func (){
    printf("Exiting from function\n");
}
int main()
{

```

```
    atexit(func);  
    printf("Exiting from main()\n");  
    _Exit(0);  
}
```

OUTPUT

```
gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ gcc exit.c -o 2_1  
gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ ./2_1  
Exiting from main()  
Exiting from function  
gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ gcc _exit.c -o 2_2  
gcc: error: 2_2: No such file or directory  
gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ gcc _exit.c -o 2_2  
gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ ./2_2  
Exiting from main()  
gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$
```

1. Write a program to demonstrate the use of wait(), waitpid() and waitid()

Ans.

PROGRAM:

Wait() function:

```
#define _POSIX_SOURCE
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    pid_t pid;
    time_t t;
    int status;

    if ((pid = fork()) < 0)
        perror("fork() error");
    else if (pid == 0) {
        time(&t);
        printf("child (pid %d) started at %s", (int) getpid(), ctime(&t));
        sleep(5);
        time(&t);
        printf("child exiting at %s", ctime(&t));
        exit(42);
    }
    else {
        printf("parent has forked child with pid of %d\n", (int) pid);
        time(&t);
        printf("parent is starting wait at %s", ctime(&t));
        if ((pid = wait(&status)) == -1)
            perror("wait() error");
        else {
            time(&t);
```

```

printf("parent is done waiting at %s", ctime(&t));
printf("the pid of the process that ended was %d\n", (int) pid);
if (WIFEXITED(status))
    printf("child exited with status of %d\n", WEXITSTATUS(status));
else if (WIFSIGNALED(status))
    printf("child was terminated by signal %d\n",
        WTERMSIG(status));
else if (WIFSTOPPED(status))
    printf("child was stopped by signal %d\n", WSTOPSIG(status));
else puts("reason unknown for child termination");
}
}
}

```

OUTPUT:

```

gadham.srinivas@DESKTOP-P2DEFRB:~/oslab/
gadham.srinivas@DESKTOP-P2DEFRB:~/oslab$ gcc wait.c -o waitex
gadham.srinivas@DESKTOP-P2DEFRB:~/oslab$ ./waitex
parent has forked child with pid of 4803
parent is starting wait at Tue Nov  1 20:49:00 2022
child (pid 4803) started at Tue Nov  1 20:49:00 2022
child exiting at Tue Nov  1 20:49:05 2022
parent is done waiting at Tue Nov  1 20:49:05 2022
the pid of the process that ended was 4803
child exited with status of 42
gadham.srinivas@DESKTOP-P2DEFRB:~/oslab$
```

Waitpid() function:

```

#define _POSIX_SOURCE

#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    pid_t pid;
    time_t t;
    int status;
```

```
if ((pid = fork()) < 0)
    perror("fork() error");
else if (pid == 0) {
    time(&t);
    printf("child (pid %d) started at %s", (int) getpid(), ctime(&t));
    sleep(5);
    time(&t);
    printf("child exiting at %s", ctime(&t));
    exit(42);
}
else {
    printf("parent has forked child with pid of %d\n", (int) pid);
    time(&t);
    printf("parent is starting wait at %s", ctime(&t));
    if ((pid = wait(&status)) == -1)
        perror("wait() error");
    else {
        time(&t);
        printf("parent is done waiting at %s", ctime(&t));
        printf("the pid of the process that ended was %d\n", (int) pid);
        if (WIFEXITED(status))
            printf("child exited with status of %d\n", WEXITSTATUS(status));
        else if (WIFSIGNALED(status))
            printf("child was terminated by signal %d\n",
                   WTERMSIG(status));
        else if (WIFSTOPPED(status))
            printf("child was stopped by signal %d\n", WSTOPSIG(status));
        else puts("reason unknown for child termination");
    }
}
```

OUTPUT:

Terminal 1

```
child exited with status 0142
gadham.srinivas@DESKTOP-P2DEFRB:~/oslab$ ps -a1
 PID TTY      STAT   TIME COMMAND
  1 ?        S1      0:00 /init
 4558 pts/0    Ss      0:00 -bash
 4831 pts/0    R+      0:00 ps -a1
gadham.srinivas@DESKTOP-P2DEFRB:~/oslab$
```

Waitid() function:

```
#define pr_exit(n) printf("%d\n", n)

#define err_sys perror

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#include <sys/wait.h>

int main(void)

{

    pid_t pid;

    int status;

    siginfo_t info;

    if ((pid = fork()) < 0)

        err_sys("fork error");

    else if (pid == 0) /* child */

        exit(7);

    if (waitid(P_PID, pid, &info, WEXITED) != pid) /* wait for child */

        err_sys("wait error");

    pr_exit(status); /* and print its status */

    if ((pid = fork()) < 0)

        err_sys("fork error");

    else if (pid == 0) /* child */

        abort(); /* generates SIGABRT */

    if (waitid(P_PID, pid, &info, WEXITED) != pid) /* wait for child */

        err_sys("wait error");

    pr_exit(status); /* and print its status */
```

```

if ((pid = fork()) < 0)
    err_sys("fork error");

else if (pid == 0)      /* child */
    status = 0;           /* divide by 0 generates SIGFPE */

if (waitid(P_PID,pid,&info,WEXITED) != pid)    /* wait for child */
    err_sys("wait error");

pr_exit(status);

exit(0);
}

```

OUTPUT:

```

gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ ./waitid
wait error: Success
0
wait error: Success
0
wait error: Invalid argument
0
wait error: Success
0
gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ -

```

2. Write a program to demonstrate the use of wait3() and wait4()**Ans.****PROGRAM:****Wait3() function:**

```

#include <sys/wait.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/time.h>
#include <sys/resource.h>

int
main(int argc, char *argv[])
{
    pid_t cpid, w;
    int status;
    struct rusage Use;
    cpid = fork();

```

```

if (cpid == -1) {
    perror("fork");
    exit(EXIT_FAILURE);
}

if (cpid == 0) /* Code executed by child */
{
    printf("Child PID is %ld\n", (long) getpid());
    if (argc == 1)
        pause(); /* Wait for signals */
    _exit(atoi(argv[1]));
} else /* Code executed by parent */
{
    do {
        w = wait3(&status, WUNTRACED | WCONTINUED, &Use);
        if (w == -1) {
            perror("waitpid");
            exit(EXIT_FAILURE);
        }
        if (WIFEXITED(status)) {
            printf("exited, status=%d\n", WEXITSTATUS(status));
        } else if (WIFSIGNALED(status)) {
            printf("killed by signal %d\n", WTERMSIG(status));
        } else if (WIFSTOPPED(status)) {
            printf("stopped by signal %d\n", WSTOPSIG(status));
        } else if (WIFCONTINUED(status)) {
            printf("continued\n");
        }
    } while (!WIFEXITED(status) && !WIFSIGNALED(status));
    exit(EXIT_SUCCESS);
}

```

OUTPUT:**Terminal 1:**

```

gadham.srinivas@DESKTOP-P2DEFRB:~/oslab$ ./wait3
Child PID is 4864

```

Terminal 2:

```

gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ ps -a1
 PID TTY      STAT   TIME COMMAND
  1 ?        S1      0:00 /init
 4558 pts/0    Ss      0:00 -bash
 4861 pts/0    T       0:00 ./wait3
 4862 pts/0    T       0:00 ./wait3
 4863 pts/0    T       0:00 ./wait3
 4864 pts/0    T       0:00 ./wait3
 4865 pts/0    R+      0:00 ps -a1
gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ kill 4864
gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ ps -a1
 PID TTY      STAT   TIME COMMAND
  1 ?        S1      0:00 /init
 4558 pts/0    Ss      0:00 -bash
 4861 pts/0    T       0:00 ./wait3
 4862 pts/0    T       0:00 ./wait3
 4863 pts/0    T       0:00 ./wait3
 4864 pts/0    T       0:00 ./wait3
 4866 pts/0    R+      0:00 ps -a1
gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ -

```

Wait4() function:**PROGRAM:**

```

#include <sys/wait.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/time.h>
#include <sys/resource.h>

int
main(int argc, char *argv[])
{
    pid_t cpid, w;
    int status;
    struct rusage Use;

    cpid = fork();
    if (cpid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

```

}

```

if (cpid == 0) {          /* Code executed by child */

    printf("Child PID is %ld\n", (long) getpid());

    if (argc == 1)

        pause();           /* Wait for signals */

    _exit(atoi(argv[1]));

} else {                  /* Code executed by parent */

    do {

        w = wait4(cpid, &status, WUNTRACED | WCONTINUED, &Use);

        if (w == -1) {

            perror("waitpid");

            exit(EXIT_FAILURE);

        }

S

        if (WIFEXITED(status)) {

            printf("exited, status=%d\n", WEXITSTATUS(status));

        } else if (WIFSIGNALED(status)) {

            printf("killed by signal %d\n", WTERMSIG(status));

        } else if (WIFSTOPPED(status)) {

            printf("stopped by signal %d\n", WSTOPSIG(status));

        } else if (WIFCONTINUED(status)) {

            printf("continued\n");

        }

    } while (!WIFEXITED(status) && !WIFSIGNALED(status));

    exit(EXIT_SUCCESS);

}
}

```

OUTPUT:**Terminal 1**

```
gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ gcc wait4.c -o wait4
gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ ./wait4
Child PID is 4881
```

Terminal 2

```
gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ ps -a1
 PID TTY      STAT   TIME COMMAND
  1 ?      S1      0:00 /init
 4558 pts/0    Ss      0:00 -bash
 4861 pts/0    T      0:00 ./wait3
 4862 pts/0    T      0:00 ./wait3
 4863 pts/0    T      0:00 ./wait3
 4864 pts/0    T      0:00 ./wait3
 4880 pts/0    T      0:00 ./wait4
 4881 pts/0    T      0:00 ./wait4
 4885 pts/0    R+      0:00 ps -a1

gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ kill 4881
gadhamsrinivas@DESKTOP-P2DEFRB:~/oslab$ ps -a1
 PID TTY      STAT   TIME COMMAND
  1 ?      S1      0:00 /init
 4558 pts/0    Ss      0:00 -bash
 4861 pts/0    T      0:00 ./wait3
 4862 pts/0    T      0:00 ./wait3
 4863 pts/0    T      0:00 ./wait3
 4864 pts/0    T      0:00 ./wait3
 4880 pts/0    T      0:00 ./wait4
 4881 pts/0    T      0:00 ./wait4
 4886 pts/0    R+      0:00 ps -a1
```