

CSE2005
Operating Systems
Assignment-1

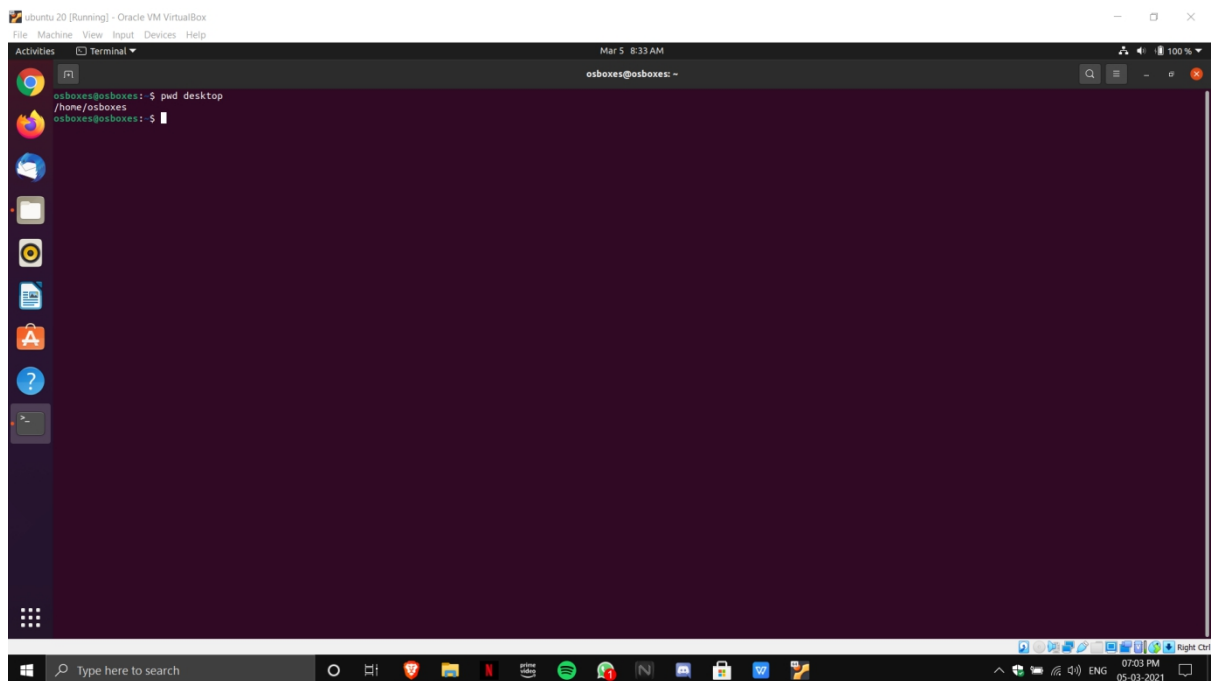
Name: Ripunjay Narula

Registration No.: 19BCE0470

Slot: F1

- (a) Shell Programming
- Handling the command line arguments

Output:



The screenshot shows a terminal window titled 'ubuntu 20 [Running] - Oracle VM VirtualBox'. The terminal prompt is 'osboxes@osboxes: ~'. The user has entered the command 'pwd desktop' and the output is '/home/osboxes'. The terminal window is running on a desktop environment with a dark theme and a sidebar on the left containing various application icons. The system tray at the bottom shows the date and time as '07:03 PM 05-03-2021'.

```
osboxes@osboxes: ~  
osboxes@osboxes:~$ pwd desktop  
/home/osboxes  
osboxes@osboxes:~$
```

```
ubuntu 20 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Mar 5 12:41
ripunjaynarula@osboxes: /home
ripunjaynarula@osboxes:~/home$ cd ..
ripunjaynarula@osboxes:~$ cd /home
ripunjaynarula@osboxes:~/home$ ls
osboxes ripunjaynarula
ripunjaynarula@osboxes:~/home$ ls -lr
ripunjaynarula@osboxes:~/home$ ls -al
total 16
drwxr-xr-x  4 root    root      4096 Mar  5 18:07 .
drwxr-xr-x 20 root    root      4096 Feb 27 03:54 ..
drwxr-xr-x 21 osboxes osboxes   4096 Mar  5 09:55 osboxes
drwxr-xr-x 19 ripunjaynarula ripunjaynarula 4096 Mar  5 12:29 ripunjaynarula
ripunjaynarula@osboxes:~/home$ find thread.c
find: 'thread.c': No such file or directory
ripunjaynarula@osboxes:~/home$ history
 1 vt
 2 vt oslab.sh
 3 chnod +x oslab
 4 cd /home/ripunjaynarula
 5 chnod +x oslab
 6 chnod +x /oslab
 7 pwd
 8 ./oslab
 9 ./oslab.sh
10 reset
11 ./oslab.sh
12 vt oslabif.sh
13 ./oslabif.sh
14 reset
15 ./oslabif.sh
16 reset
17 ./oslabif.sh
18 vt oslabpc.sh
19 vt oslabpc.c
20 gcc oslabpc.c
21 ./a.out
22 reset
23 gcc oslabpc.c
24 ./a.out
25 gcc oslabpc.c
26 reset
27 gcc oslabpc.c
28 ./a.out
29 reset
30 vt file1.c
31 gcc file1.c
32 vt file2.c
33 gcc file2.c
34 gcc file2.c
35 ./a.out
36 gcc file2.c
37 ./a.out
38 vt oslabx.c
39 gcc oslabx.c
40 reset
41 gcc oslabx.c
42 ./a.out
43 reset
44 gcc oslabx.c
45 ./a.out
46 reset
```

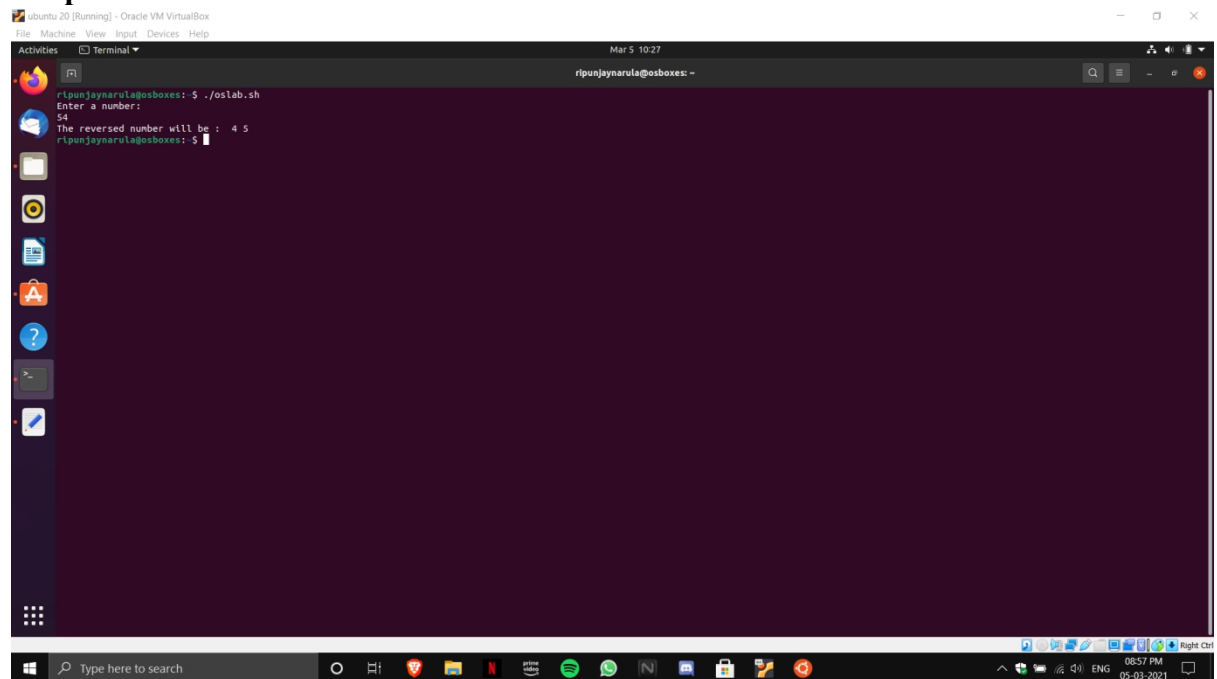
```
ubuntu 20 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Mar 5 12:43
ripunjaynarula@osboxes: /home
ripunjaynarula@osboxes:~/home$ hostname
osboxes
ripunjaynarula@osboxes:~/home$ history
 1 vt
 2 vt oslab.sh
 3 chnod +x oslab
 4 cd /home/ripunjaynarula
 5 chnod +x oslab
 6 chnod +x /oslab
 7 pwd
 8 ./oslab
 9 ./oslab.sh
10 reset
11 ./oslab.sh
12 vt oslabif.sh
13 ./oslabif.sh
14 reset
15 ./oslabif.sh
16 reset
17 ./oslabif.sh
18 vt oslabpc.sh
19 vt oslabpc.c
20 gcc oslabpc.c
21 ./a.out
22 reset
23 gcc oslabpc.c
24 ./a.out
25 gcc oslabpc.c
26 reset
27 gcc oslabpc.c
28 ./a.out
29 reset
30 vt file1.c
31 gcc file1.c
32 vt file2.c
33 gcc file2.c
34 gcc file1.c
35 ./a.out
36 gcc file2.c
37 ./a.out
38 vt oslabx.c
39 gcc oslabx.c
40 reset
41 gcc oslabx.c
42 ./a.out
43 reset
44 gcc oslabx.c
45 ./a.out
46 reset
```

- String reversal

Code:

```
echo "Enter a number: "
read str
len=${#str}
rev=""
for((i=0;i<len;i++))
do rev="$rev ${str:(n-1-i):1}"
done
echo "The reversed number will be : $rev"
```

Output:



The screenshot shows a terminal window titled "Terminal" with the user "ripunjaynarula@osboxes: ~". The terminal displays the following output:

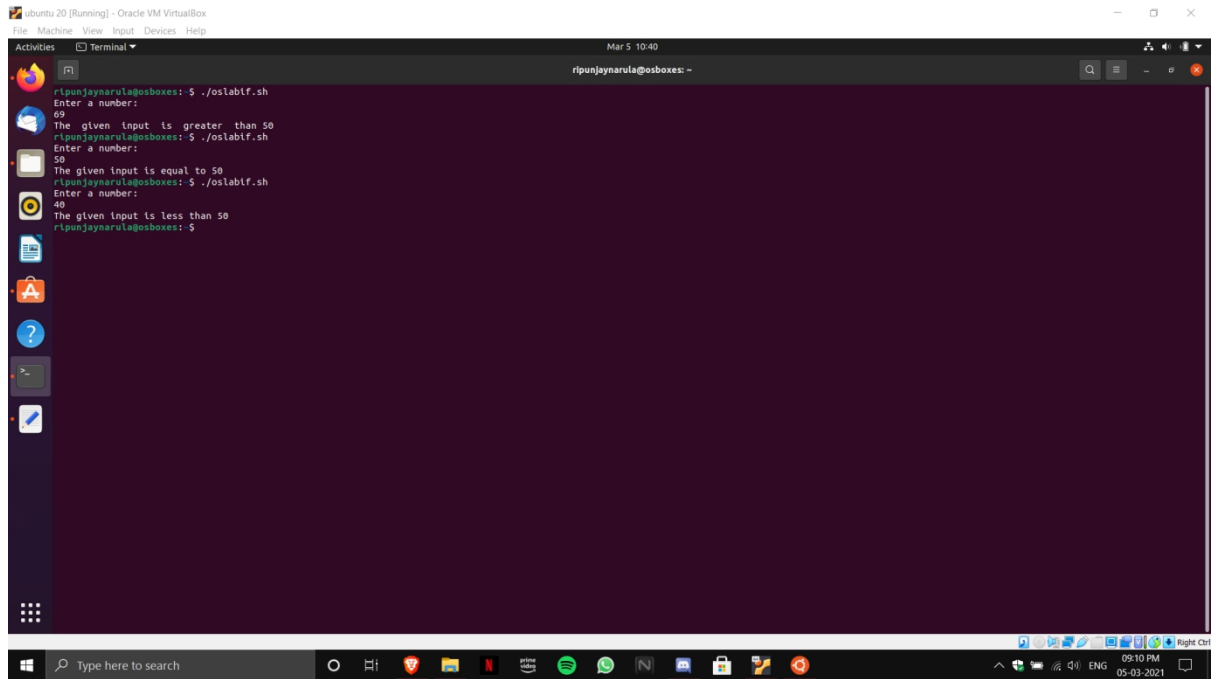
```
ripunjaynarula@osboxes:~$ ./oslab.sh
Enter a number:
54
The reversed number will be : 4 5
ripunjaynarula@osboxes:~$
```

- If-Else, Nested If Else, Switch cases in shell

Code:

If-Else, Nested If Else

```
echo "Enter a number: "
read n
if [ $n -gt 50 ]
then
echo "The given input is greater than 50"
elif [ $n == 50 ]
then
echo "The given input is equal to 50"
else
echo "The given input is less than 50"
fi
```

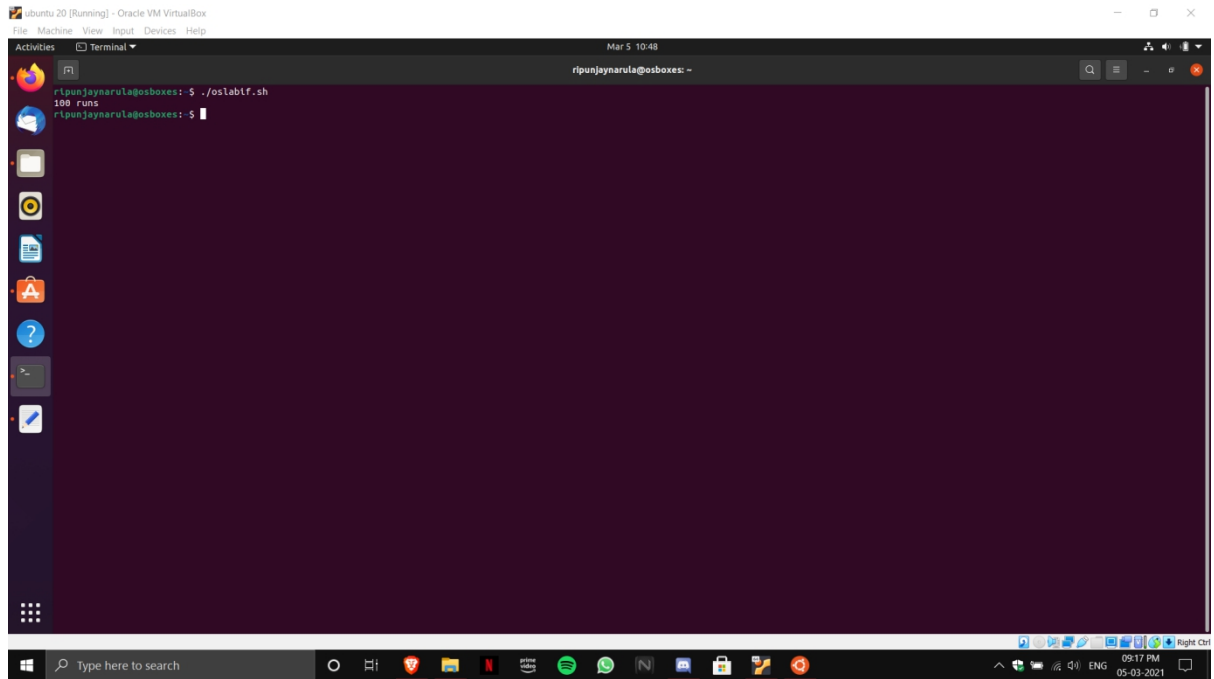


```
ubuntu 20 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Mar 5 10:40
ripunjaynarula@osboxes: ~
ripunjaynarula@osboxes: $ ./oslabtf.sh
Enter a number:
69
The given input is greater than 50
ripunjaynarula@osboxes: $ ./oslabtf.sh
Enter a number:
50
The given input is equal to 50
ripunjaynarula@osboxes: $ ./oslabtf.sh
Enter a number:
40
The given input is less than 50
ripunjaynarula@osboxes: $
```

Switch case

```
player="player1"
case "$player" in "player2")
echo "70 runs"
;; "player3")
echo "150 runs"
;; "player1")
echo "100 runs"
;;
esac
```

Output:



- (b) Parent child process creation using fork() and exec() system call

Checking the Process Identifier

Assigning new task to child

Providing the path name and program name to exec()

Synchronizing Parent and child process using wait()

Code:

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

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

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

```
int main()
```

```
{
```

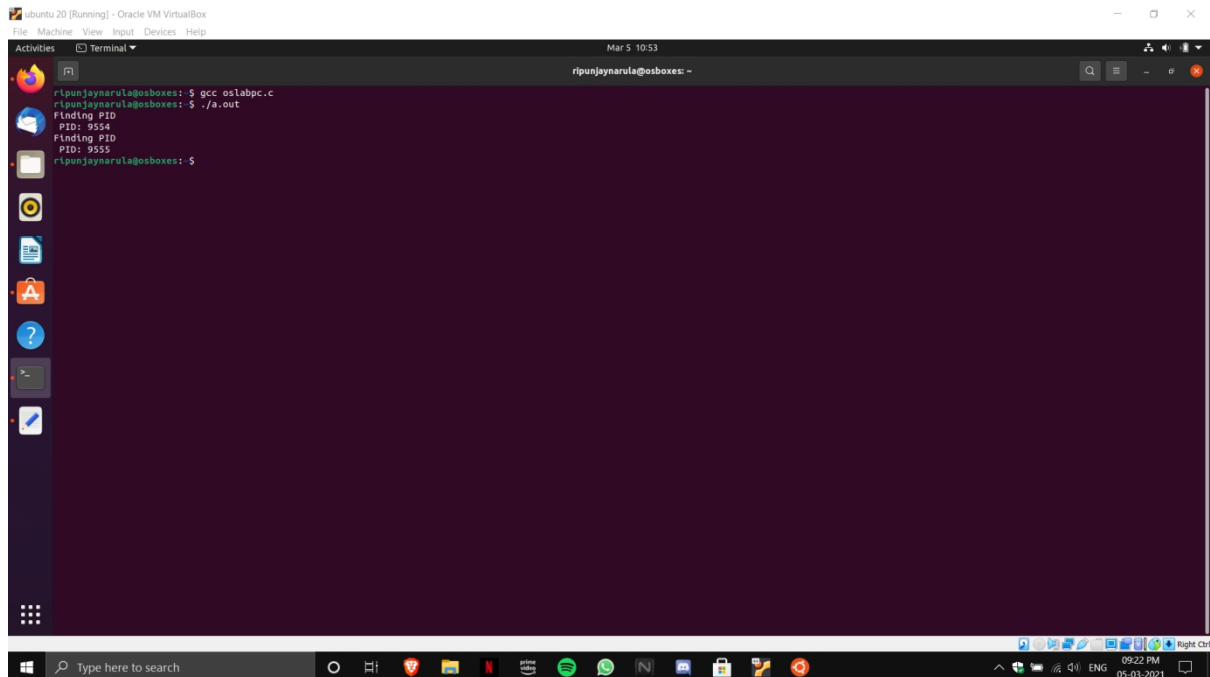
```
fork();
```

```
printf("Finding PID\n");
```

```
printf(" PID: %d\n",getpid());
```

```
return 0;
```

```
}
```



File1.c:

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
printf("PID of file1.c = %d\n",getpid());
char*args[]= {"Operating Systems",NULL};
execv("./file2",args);
return 0;
}
```

File2.c:

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

```

#include<unistd.h>

#include<stdlib.h>

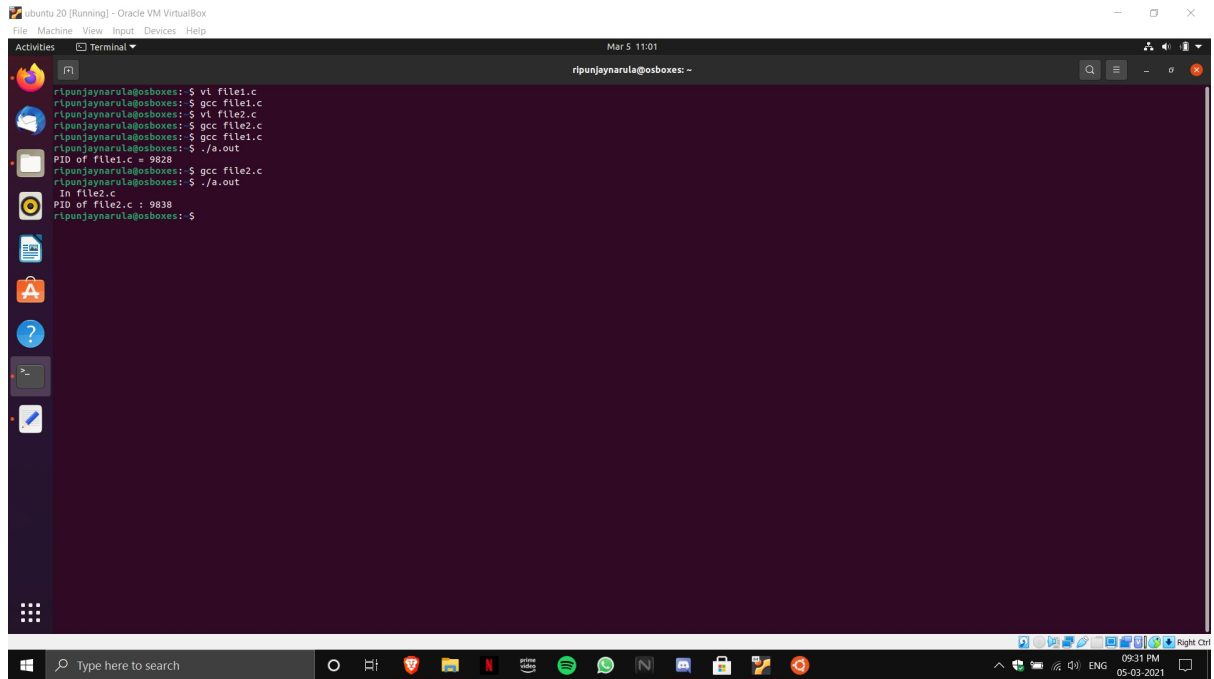
int main(){

printf(" In file2.c\n");

printf("PID of file2.c : %d \n",getpid());

return 0; }

```



```

rtpunjaynarula@osboxes:~$ vi file1.c
rtpunjaynarula@osboxes:~$ gcc file1.c
rtpunjaynarula@osboxes:~$ vi file2.c
rtpunjaynarula@osboxes:~$ gcc file2.c
rtpunjaynarula@osboxes:~$ gcc file1.c
rtpunjaynarula@osboxes:~$ ./a.out
PID of file1.c = 9828
rtpunjaynarula@osboxes:~$ gcc file2.c
rtpunjaynarula@osboxes:~$ ./a.out
In file2.c
PID of file2.c : 9838
rtpunjaynarula@osboxes:~$

```

(With Wait)

```

#include<stdio.h>

#include<sys/types.h>

#include <sys/wait.h>

#include<unistd.h>

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

    int id= fork();

    int n,i;

    if(id==0){

        printf("\nParent Process: %d\n",getpid());

        n=1; }

    else{

```

```

printf("\n Child Process: %d \n",getpid());

n=5; }

printf("\n");

if(id!=0){

pid_t wait(int *id); }

for(i=n;i<n+5;i++){

printf("%d ",n); }

if(id!=0){

printf("\n"); }

return 0; }

```

```

ubuntu 20 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
ripunjaynarula@osboxes: ~
ripunjaynarula@osboxes: $ gcc oslabx.c
ripunjaynarula@osboxes: $ ./a.out
Child Process: 10378
$ $ $ $ $
Parent Process: 10379
1 1 1 1 1 ripunjaynarula@osboxes: $

```

(Without Wait)

```

#include<stdio.h>
#include<sys/types.h>
#include <sys/wait.h>
#include<unistd.h>
int main(int argc , char *argv[]){
    int id= fork();
    int n,i;
    if(id==0){
        printf("\nParent Process: %d\n",getpid());
    }
}

```



```

n=1; }
else{
printf("\n Child Process: %d \n",getpid());
n=5; }
printf("\n");

for(i=n;i<n+5;i++){
printf("%d ",n); }
if(id!=0){
printf("\n"); }
return 0;

```

The screenshot shows a terminal window titled 'ubuntu 20 [Running] - Oracle VM VirtualBox'. The user 'ripunjaynarula@osboxes' has executed the command 'gcc oslabx.c' and then './a.out'. The output of the program is as follows:

```

Child Process: 10425
5 5 5 5 5
Parent Process: 10426
1 1 1 1 1
ripunjaynarula@osboxes: $

```

(c) Process and Thread Management Write a program to create a thread and perform the following (Easy)

- Create a thread runner function
- Set the thread attributes
- Join the parent and thread
- Wait for the thread to complete

Code:

```

#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
void *entry_point(void*value){
printf("Second Thread\n");

```

```

int num=(int) value;
printf("%d is the number\n",*num);
}
int main(int argc, char **argv){
pthread_t thread;
printf("Thread 1 \n");
int num=777;
pthread_create(&thread,NULL, entry_point,&num);
pthread_join(thread, NULL);
pid_t wait();
printf("First thread");
return SUCCESS; }

```

Output:

```

r@punjaynarula@osboxes: $ gcc -pthread -o thread1 thread1.c
r@punjaynarula@osboxes: $ ./thread1
Thread 1
Second Thread
777 is the number
First thread:r@punjaynarula@osboxes: $

```

- (d) Write a program to create a thread to find the factorial of a natural number 'n'.
(Medium)

Code:

```

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <sys/types.h>

#include <sys/wait.h>

void *factorial(void *value){

```

```

printf("Now in Factorial Thread\n");
int *num=(int *) value;
int n= *num;
int i,ans=1;
for(i=1;i<=n;i++){
ans=ans*i;
}
if(n==0){
ans=1;
}
printf("The factorial is : %d\n",ans);

}

int main(int argc , char **argv){
pthread_t thread;
int num;
printf("Enter the number: ");
scanf("%d",&num);

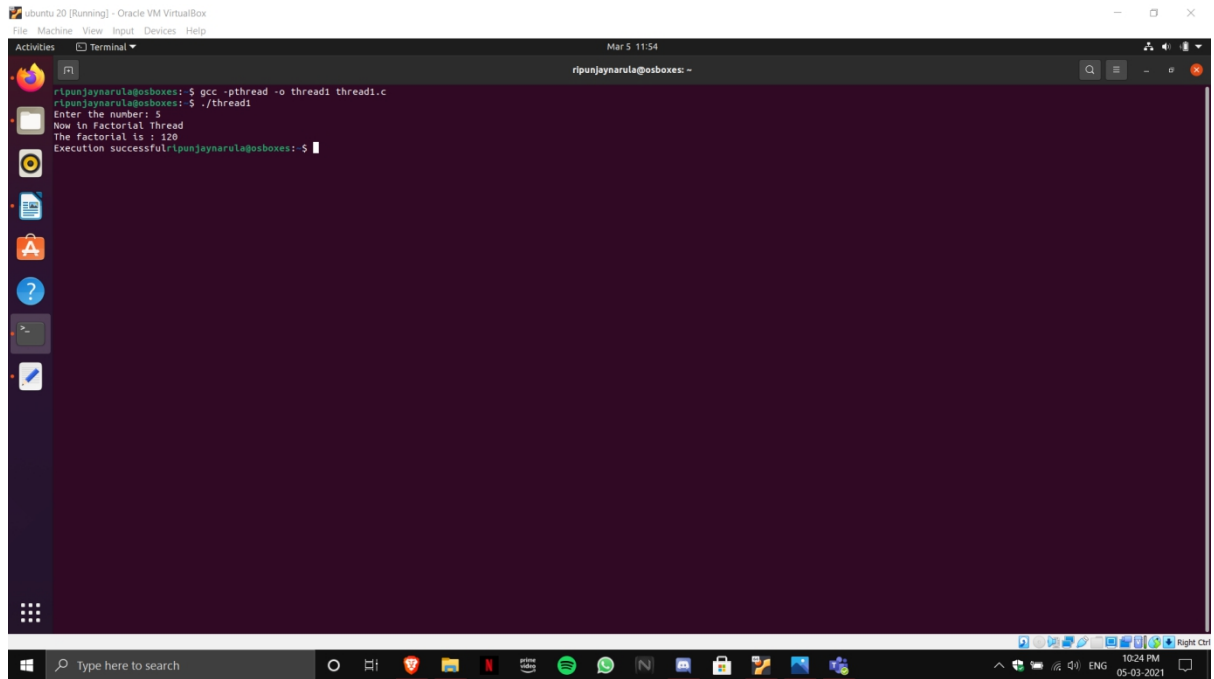
pthread_create(&thread,NULL, factorial ,&num);
pthread_join(thread, NULL);

pid_t wait(int *num);

printf("SUCCESS");
return 0;
}

```

Output:



- (e) Assume that two processes named client and server running in the system. It is required that these two processes should communicate with each other using shared memory concept. The server writes alphabets from a..z to the shared memory .the client should read the alphabets from the shared memory and convert it to A...Z. Write a program to demonstrate the above mentioned scenario. (Medium)

Code:

Client.c

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <stdlib.h>
#define SHMSZ 27
main()
{
    int shmid,i;
    key_t key;
    char *shm, *s;
    printf("\nClient Running\n");
    key = 5678;
    if ((shmid = shmget(key, SHMSZ, 0666)) < 0) {
        perror("shmget");
        exit(1);
    }
```

```

}
if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
perror("shmat");
exit(1);
}
for (s = shm; *s!=NULL; s++)
putchar(*s);
putchar('\n');
for(i=65;i<=90;i++){
*shm = (char)i;
shm++;
}
exit(0);
}

```

```

ubuntu 20 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Activities Terminal Mar 5 12:18
ripunjaynarula@osboxes: ~

ripunjaynarula@osboxes:~$ gcc client.c
client.c:7:11: warning: return type defaults to 'int' [-Wimplicit-int]
7 | main()
  | ^~~~~
client.c: In function 'main':
client.c:22:11: warning: comparison between pointer and integer
22 | for (s = shm; *s!=NULL; s++)
  |           ^~
ripunjaynarula@osboxes:~$ ./a.out
Client Running
shmgset: No such file or directory
ripunjaynarula@osboxes:~$

```

Server.c

```

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#define SHMSZ 27
main()
{
char c;
int shmid;
key_t key;
char *shm, *s;

```

```

printf("\n19BCE0470\n");
printf("\nServer Running\n");
key = 5678;
if ((shm = shmget(key, SHMSZ, IPC_CREAT | 0666)) < 0) {
perror("shmget");
exit(1);
}
if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
perror("shmat");
exit(1);
}
s = shm;
for (c = 'a'; c <= 'z'; c++)
*s++ = c;
*s = NULL;
while (shm != "")
sleep(1);
exit(0);
}

```

- (f) The Collatz conjecture concerns what happens when we take any positive integer n and apply the following algorithm: $n = n/2$, if n is even $n = 3 \times n + 1$, if n is odd. The conjecture states that when this algorithm is continually applied, all positive integers will eventually reach 1. For example, if $n = 35$, the sequence is 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1. Write a C program using the `fork()` system call that generates this sequence in the child process. The starting number will be provided from the command line. For example, if 8 is passed as a parameter on the Command line, the child process will output 8, 4, 2, 1. Because the parent and child processes have their own copies of the data, it will be necessary for the child to output the sequence. Have the parent invoke the `wait()` call to wait for the child process to complete before exiting the program (High).

Code:

```

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#define SHMSZ 27

```

```

main()
{
char c;
int shmid;
key_t key;
char *shm, *s;
printf("\n19BCE0470\n");
printf("\nServer Running\n");
key = 5678;
if ((shmid = shmget(key, SHMSZ, IPC_CREAT | 0666)) < 0) {
perror("shmget");
exit(1);
}
if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
perror("shmat");
exit(1);
}
s = shm;
for (c = 'a'; c <= 'z'; c++)
*s++ = c;
*s = NULL;
while (shm != "")
sleep(1);
exit(0);
}

```

Output:

```

ubuntu 20 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Mar 5 12:24
ripunjaynarula@osboxes: ~
ripunjaynarula@osboxes: $ gcc os.c
os.c: In function 'main':
os.c:33:11: warning: implicit declaration of function 'wait' [-Wimplicit-function-declaration]
33 |     wait();
    |     ^~~~~
ripunjaynarula@osboxes: $ ./a.out
19BCE0470
Please enter a number greater >0 to find the CollatzConjecture.
5
Parents is waiting on child
Child is working
5
10
8
4
2
1
Child process completed.
Parent process is done.
ripunjaynarula@osboxes: $

```

g) Write a multithreaded program that calculates various statistical values for a list of numbers. This program will be passed a series of numbers on the command line and will then create three separate worker threads. One thread will determine the average of the numbers, the second will determine the maximum value, and the third will determine the minimum value. For example, suppose your program is passed the integers 90 81 78 95 79 72 85 , the program will report the average value as 82. The minimum value as 72. The maximum value as 95. The variables representing the average, minimum, and maximum values will be stored globally. The worker threads will set these values, and the parent thread will output the values once the workers have exited. (High)

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/wait.h>
int a[100],n,i;
void *th()
{
    int sum=0;
    float avg;
    printf("19BCE0470");
    printf("enter number of inputs: ");
    scanf("%d",&n);
    printf("enter inputs: ");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(i=0;i<n;i++)
    {
        sum=sum+a[i];
    }
    avg=sum/n;
    printf("The Average value is:%f",avg);
}
void *t1()
{
    int c=a[0];
    for(int i=1;i<n;i++)
    {
        if(c>a[i])
        {
            c=a[i];
        }
    }
}
```



```

printf("\nThe Minimum value is:=%d",c);
}
void *t2()
{
int s=a[0];
for(int i=1;i<n;i++)
{
if(s<a[i])
{s=a[i];
}
}
printf("\nThe Maximum value is:=%d",s);
}
int main()
{
int n,i;
pthread_t tr1;
pthread_t tr2;
pthread_t tr3;
n=pthread_create(&tr1,NULL,&th,NULL);
pthread_join(tr1,NULL);
n=pthread_create(&tr2,NULL,&t1,NULL);
pthread_join(tr2,NULL);
n=pthread_create(&tr3,NULL,&t2,NULL);
pthread_join(tr3,NULL);
}

```

Output:

