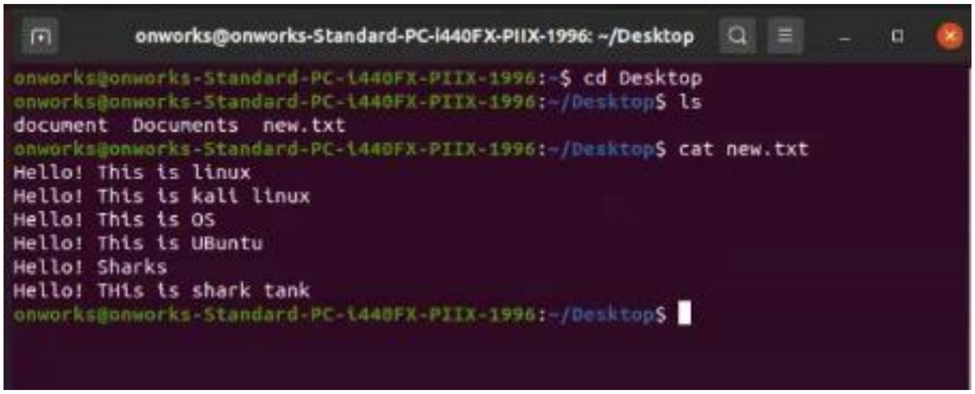
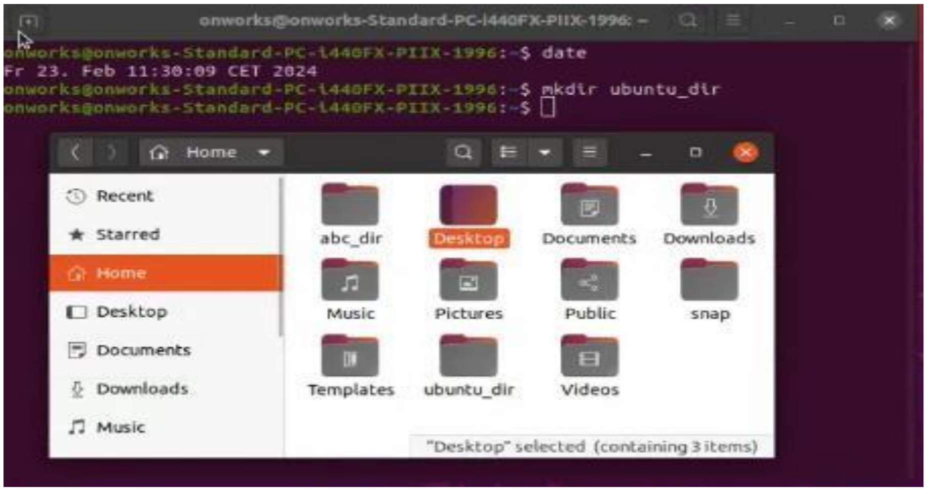
1. Explore basic LINUX commands like mkdir, chdir, cat, ls,chmod.

- mkdir directory_name
- chdir [directory] : Where directory is the name of the directory you want to change to. If you do not specify a directory, the chdir command will change to your home directory.
- cat FILE : The cat command reads files sequentially, displaying their content to the terminal.
- ls [file/directory] : The file/directory argument is the path to the directory whose contents you want to list. If no argument is specified, the ls command will list the contents of the current working directory.
- chmod [reference][operator][mode] [File_name] : the chmod command is used to change the access mode of a file. The name is an abbreviation of change mode.



```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ cd Desktop
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls
document  Documents  new.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ cat new.txt
Hello! This is linux
Hello! This is kali linux
Hello! This is OS
Hello! This is UBuntu
Hello! Sharks
Hello! THis is shark tank
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ 
```



```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ date
Fr 23. Feb 11:30:09 CET 2024
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ mkdir ubuntu_dir
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ 
```

2. Explore basic LINUX commands like pwd,cd, mv, cp, rm.

- Pwd : The 'pwd' (print working directory) command in Linux is a built-in command that displays the full pathname of the current directory
- Cd : cd ..: - This will change the current directory to the parent directory.

cd ~: - This will change the current directory to the home directory.

cd /path/to/directory: - It will change the current directory to the specified directory.

- mv [source_file_name(s)] [Destination_file_name] : to Rename a file in Linux

  mv [source_file_name(s)] [Destination_path] : to Move a File in Linux

- cp source_file destination : This command creates a copy of the `source_file` at the specified `destination`
- rm [file or directory name] : The rm command in Linux is used to delete files and directories

3. Write shell scripts to do the following:

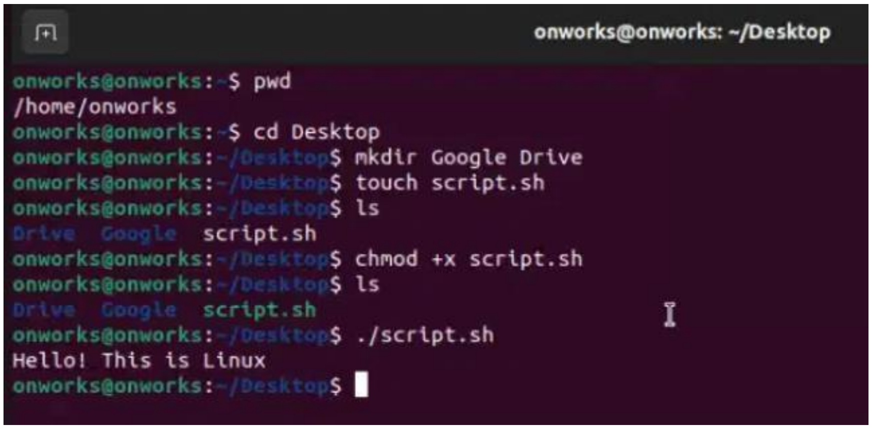
a) Display OS version, release number, kernel version

```
#!/bin/sh
Cat /etc/os-release
Lsb_release –a
Hostnamectl
Uname -r
```

b) Display top 10 processes in descending order

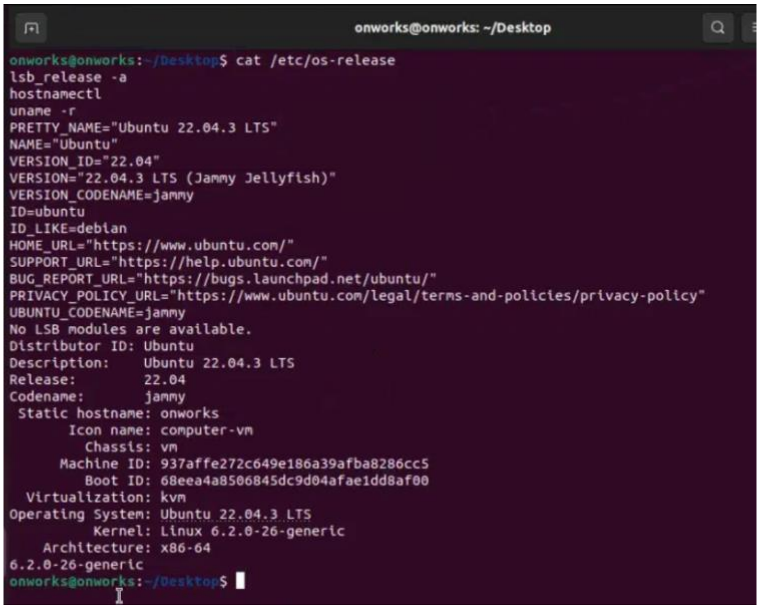
```
#!/bin/sh
$ps –eo size,command –sort –size |head
```

1. <u>Program for shell script</u>



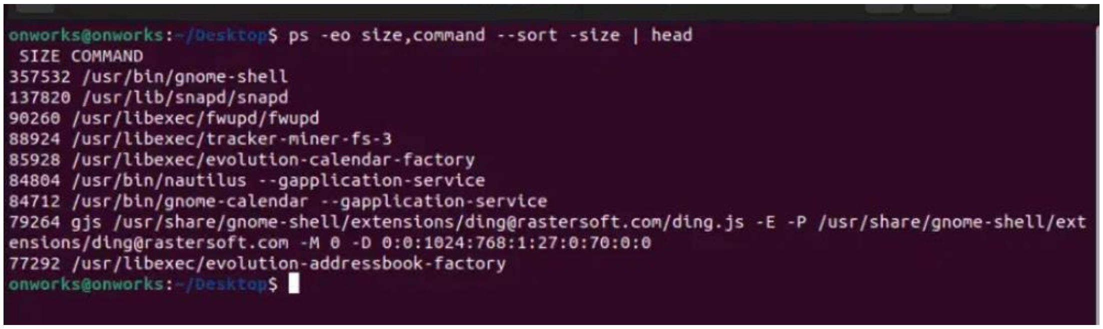
```
onworks@onworks:~$ pwd
/home/onworks
onworks@onworks:~$ cd Desktop
onworks@onworks:~/Desktop$ mkdir Google Drive
onworks@onworks:~/Desktop$ touch script.sh
onworks@onworks:~/Desktop$ ls
Drive  Google  script.sh
onworks@onworks:~/Desktop$ chmod +x script.sh
onworks@onworks:~/Desktop$ ls
Drive  Google  script.sh
onworks@onworks:~/Desktop$ ./script.sh
Hello! This is Linux
onworks@onworks:~/Desktop$
```

2. <u>Display os version,Release Number, Kernel Version</u>



```
onworks@onworks:~/Desktop$ cat /etc/os-release
lsb_release -a
hostnamectl
uname -r
PRETTY_NAME="Ubuntu 22.04.3 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.3 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.3 LTS
Release:        22.04
Codename:       jammy
 Static hostname: onworks
       Icon name: computer-vm
         Chassis: vm
      Machine ID: 937affe272c649e186a39afba8286cc5
         Boot ID: 68eea4a8506845dc9d04afae1dd8af00
  Virtualization: kvm
Operating System: Ubuntu 22.04.3 LTS
          Kernel: Linux 6.2.0-26-generic
    Architecture: x86-64
6.2.0-26-generic
onworks@onworks:~/Desktop$ 
```

3.Display Top 10 processes in descending order



```
onworks@onworks:~/Desktop$ ps -eo size,command --sort -size | head
 SIZE COMMAND
357532 /usr/bin/gnome-shell
137820 /usr/lib/snapd/snapd
90260 /usr/libexec/fwupd/fwupd
88924 /usr/libexec/tracker-miner-fs-3
85928 /usr/libexec/evolution-calendar-factory
84804 /usr/bin/nautilus --gapplication-service
84712 /usr/bin/gnome-calendar --gapplication-service
79264 gjs /usr/share/gnome-shell/extensions/ding@rastersoft.com/ding.js -E -P /usr/share/gnome-shell/ext
ensions/ding@rastersoft.com -M 0 -D 0:0:1024:768:1:27:0:70:0:0
77292 /usr/libexec/evolution-addressbook-factory
onworks@onworks:~/Desktop$ 
```

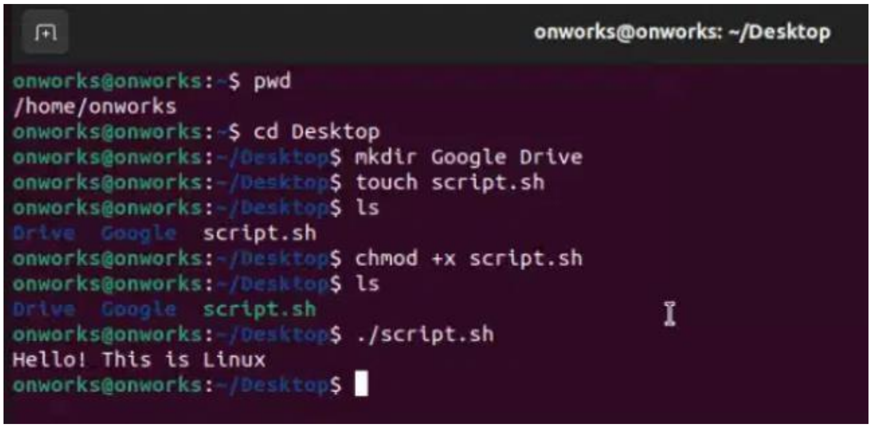
4. Write shell scripts to do the following:

a. Display processes with highest memory usage.

```
#!/bin/sh

Top –b –o +%MEM | head -17
```

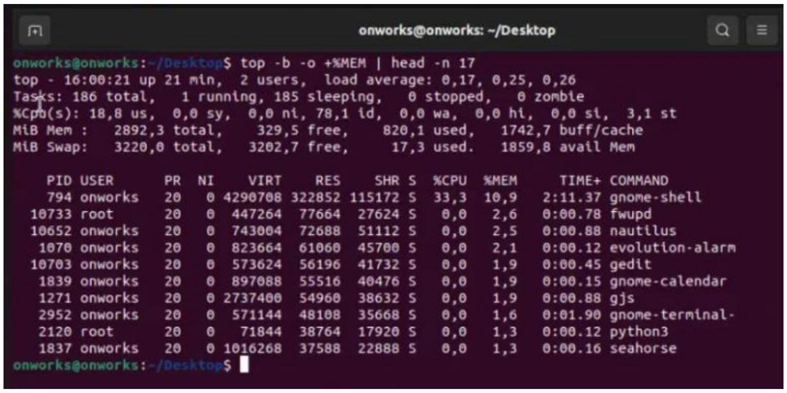
b.Display current logged in user and log name.

1. Program for shell script



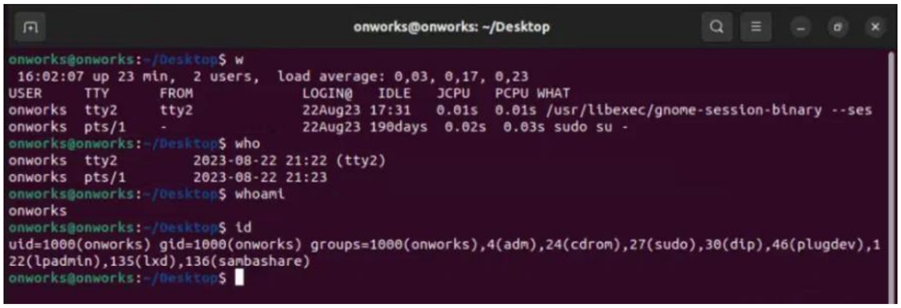
```
onworks@onworks:~$ pwd
/home/onworks
onworks@onworks:~$ cd Desktop
onworks@onworks:~/Desktop$ mkdir Google Drive
onworks@onworks:~/Desktop$ touch script.sh
onworks@onworks:~/Desktop$ ls
Drive  Google  script.sh
onworks@onworks:~/Desktop$ chmod +x script.sh
onworks@onworks:~/Desktop$ ls
Drive  Google  script.sh
onworks@onworks:~/Desktop$ ./script.sh
Hello! This is Linux
onworks@onworks:~/Desktop$ 
```

2.Display process with highest memory uses



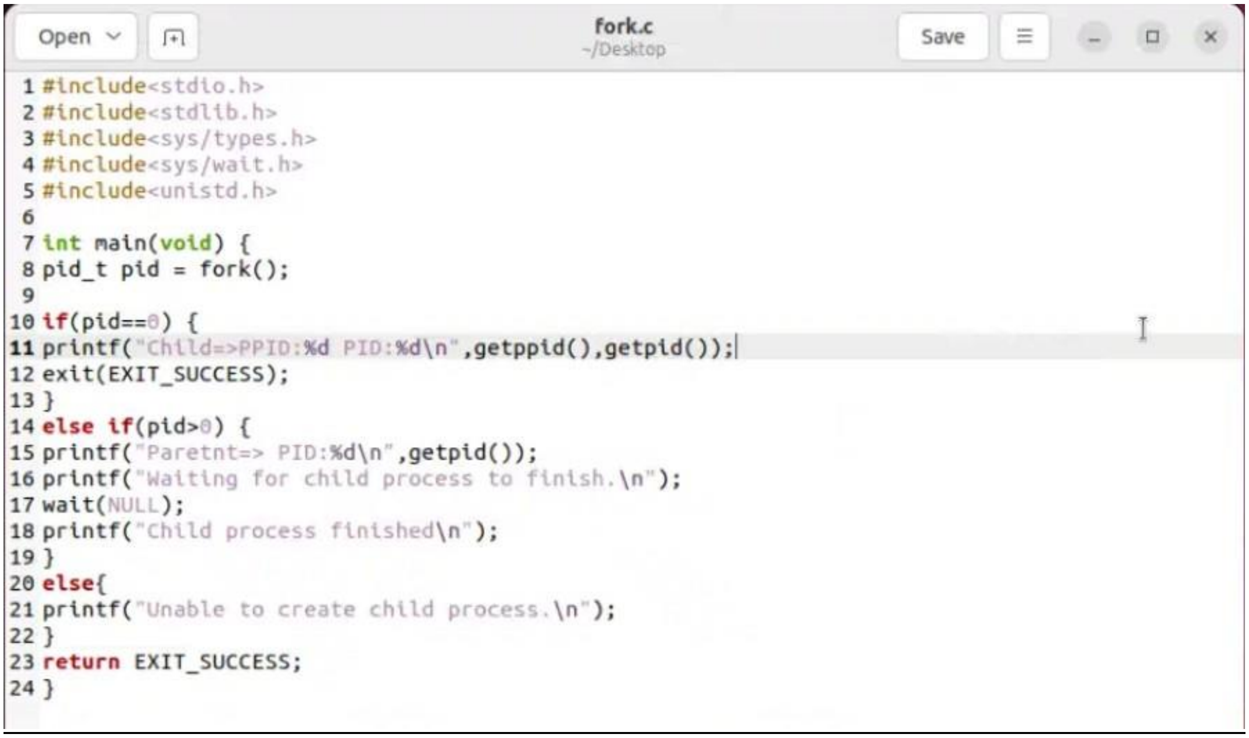
```
onworks@onworks:~/Desktop$ top -b -o +%MEM | head -n 17
top - 16:00:21 up 21 min,  2 users,  load average: 0,17, 0,25, 0,26
Tasks: 186 total,   1 running, 185 sleeping,   0 stopped,   0 zombie
%Cpu(s): 18,8 us,  0,0 sy,  0,0 ni, 78,1 id,  0,0 wa,  0,0 hi,  0,0 si,  3,1 st
MiB Mem :   2892,3 total,    329,5 free,    820,1 used,   1742,7 buff/cache
MiB Swap:   3220,0 total,   3202,7 free,     17,3 used.   1859,8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    794 onworks   20   0 4290708 322852 115172 S  33,3  10,9   2:11.37 gnome-shell
  10733 root      20   0  447264  77664  27624 S   0,0   2,6   0:00.78 fwupd
  10652 onworks   20   0  743004  72688  51112 S   0,0   2,5   0:00.88 nautilus
   1070 onworks   20   0  823664  61060  45700 S   0,0   2,1   0:00.12 evolution-alarm
  10703 onworks   20   0  573624  56196  41732 S   0,0   1,9   0:00.45 gedit
   1839 onworks   20   0  897088  55516  40476 S   0,0   1,9   0:00.15 gnome-calendar
   1271 onworks   20   0 2737400  54960  38632 S   0,0   1,9   0:00.88 gjs
   2952 onworks   20   0  571144  48108  35668 S   0,0   1,6   0:01.90 gnome-terminal-
   2120 root      20   0   71844  38764  17920 S   0,0   1,3   0:00.12 python3
   1837 onworks   20   0 1016268  37588  22888 S   0,0   1,3   0:00.16 seahorse
onworks@onworks:~/Desktop$ 
```

3. Display current logged in user and log name



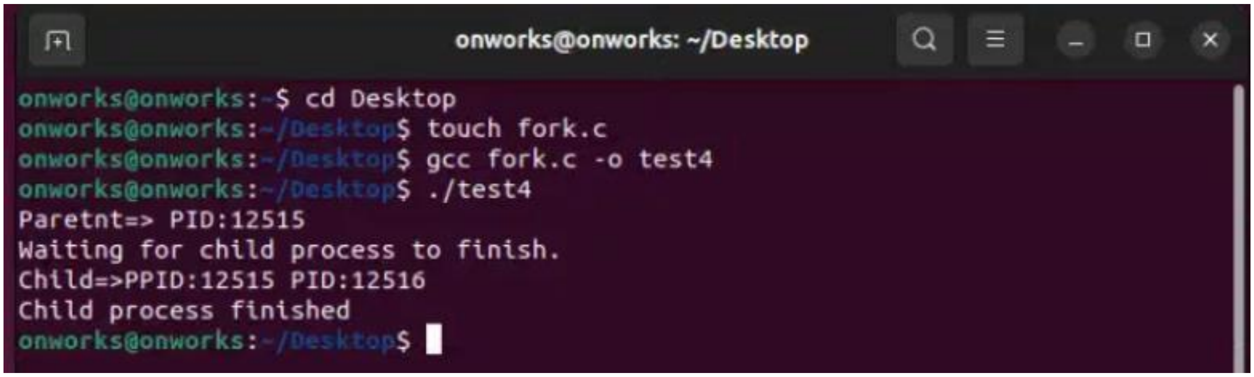
```
onworks@onworks:~/Desktop$ w
 16:02:07 up 23 min,  2 users,  load average: 0,03, 0,17, 0,23
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
onworks  tty2     tty2             22Aug23 17:31   0.01s  0.01s /usr/libexec/gnome-session-binary --ses
onworks  pts/1    -                22Aug23 190days  0.02s  0.03s sudo su -
onworks@onworks:~/Desktop$ who
onworks  tty2         2023-08-22 21:22 (tty2)
onworks  pts/1        2023-08-22 21:23
onworks@onworks:~/Desktop$ whoami
onworks
onworks@onworks:~/Desktop$ id
uid=1000(onworks) gid=1000(onworks) groups=1000(onworks),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),1
22(lpadmin),135(lxd),136(sambashare)
onworks@onworks:~/Desktop$ 
```

5. Create a child process in Linux using the fork system call. From the child process obtain the process ID of both child and parent by using getpid and getppid system call.

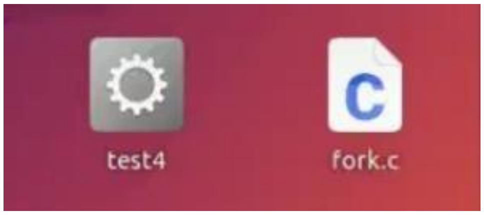


```c
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/wait.h>
#include<unistd.h>

int main(void) {
pid_t pid = fork();

if(pid==0) {
printf("Child=>PPID:%d PID:%d\n",getppid(),getpid());
exit(EXIT_SUCCESS);
}
else if(pid>0) {
printf("Paretnt=> PID:%d\n",getpid());
printf("Waiting for child process to finish.\n");
wait(NULL);
printf("Child process finished\n");
}
else{
printf("Unable to create child process.\n");
}
return EXIT_SUCCESS;
}
```

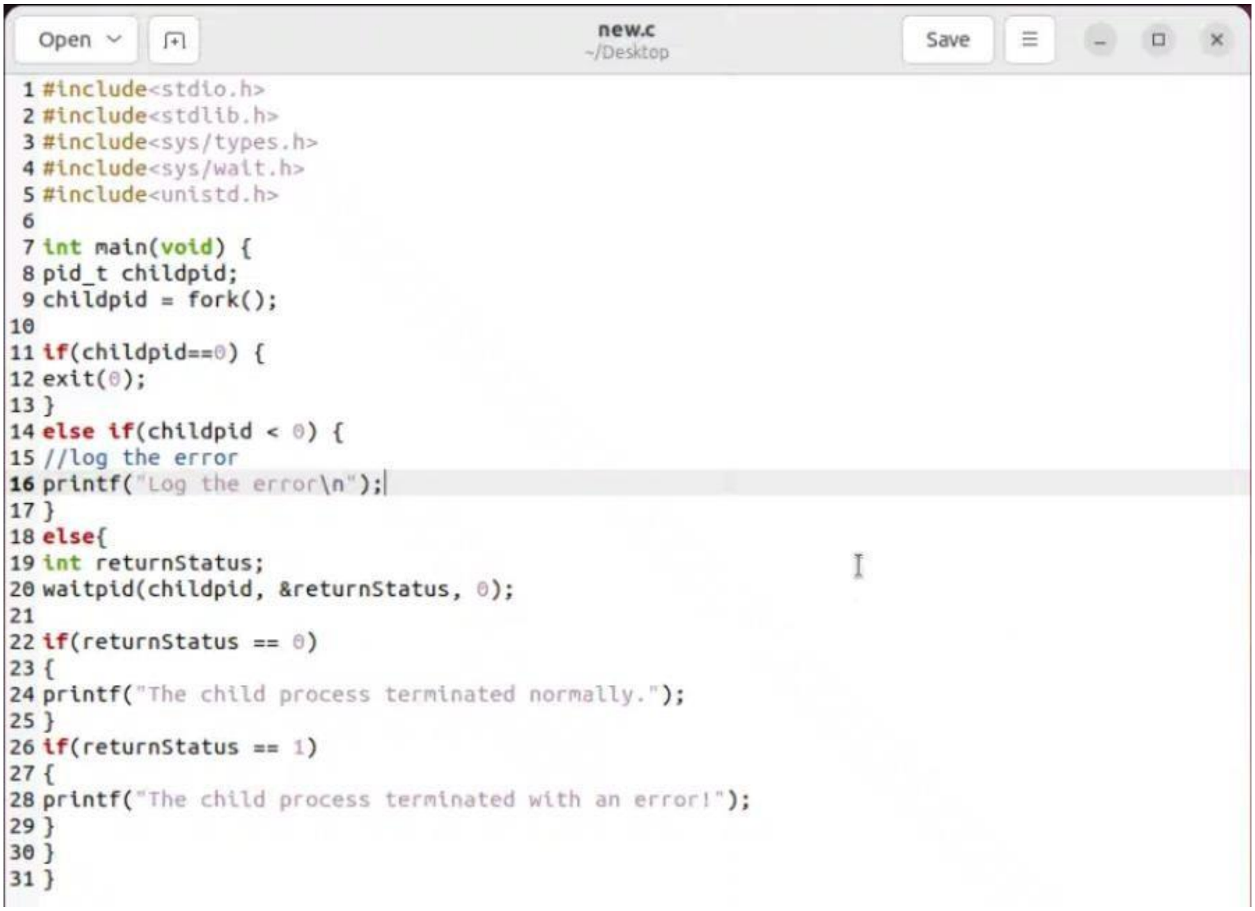
********************** OUTPUT **********************



```
onworks@onworks:~$ cd Desktop
onworks@onworks:~/Desktop$ touch fork.c
onworks@onworks:~/Desktop$ gcc fork.c -o test4
onworks@onworks:~/Desktop$ ./test4
Paretnt=> PID:12515
Waiting for child process to finish.
Child=>PPID:12515 PID:12516
Child process finished
onworks@onworks:~/Desktop$
```

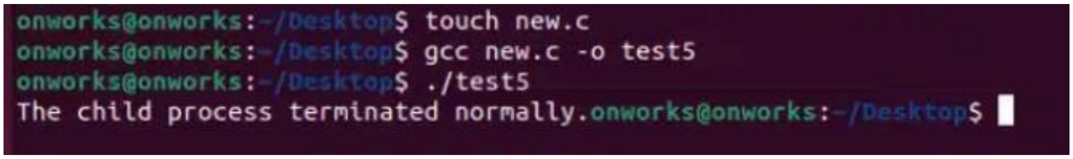


6. Explore wait and waitpid before termination of process.

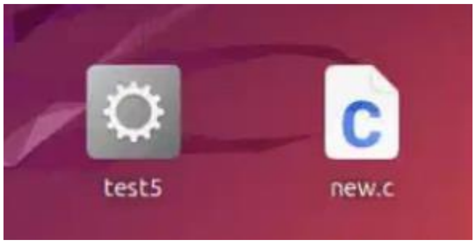


```c
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/wait.h>
#include<unistd.h>

int main(void) {
pid_t childpid;
childpid = fork();

if(childpid==0) {
exit(0);
}
else if(childpid < 0) {
//log the error
printf("Log the error\n");
}
else{
int returnStatus;
waitpid(childpid, &returnStatus, 0);

if(returnStatus == 0)
{
printf("The child process terminated normally.");
}
if(returnStatus == 1)
{
printf("The child process terminated with an error!");
}
}
}
```

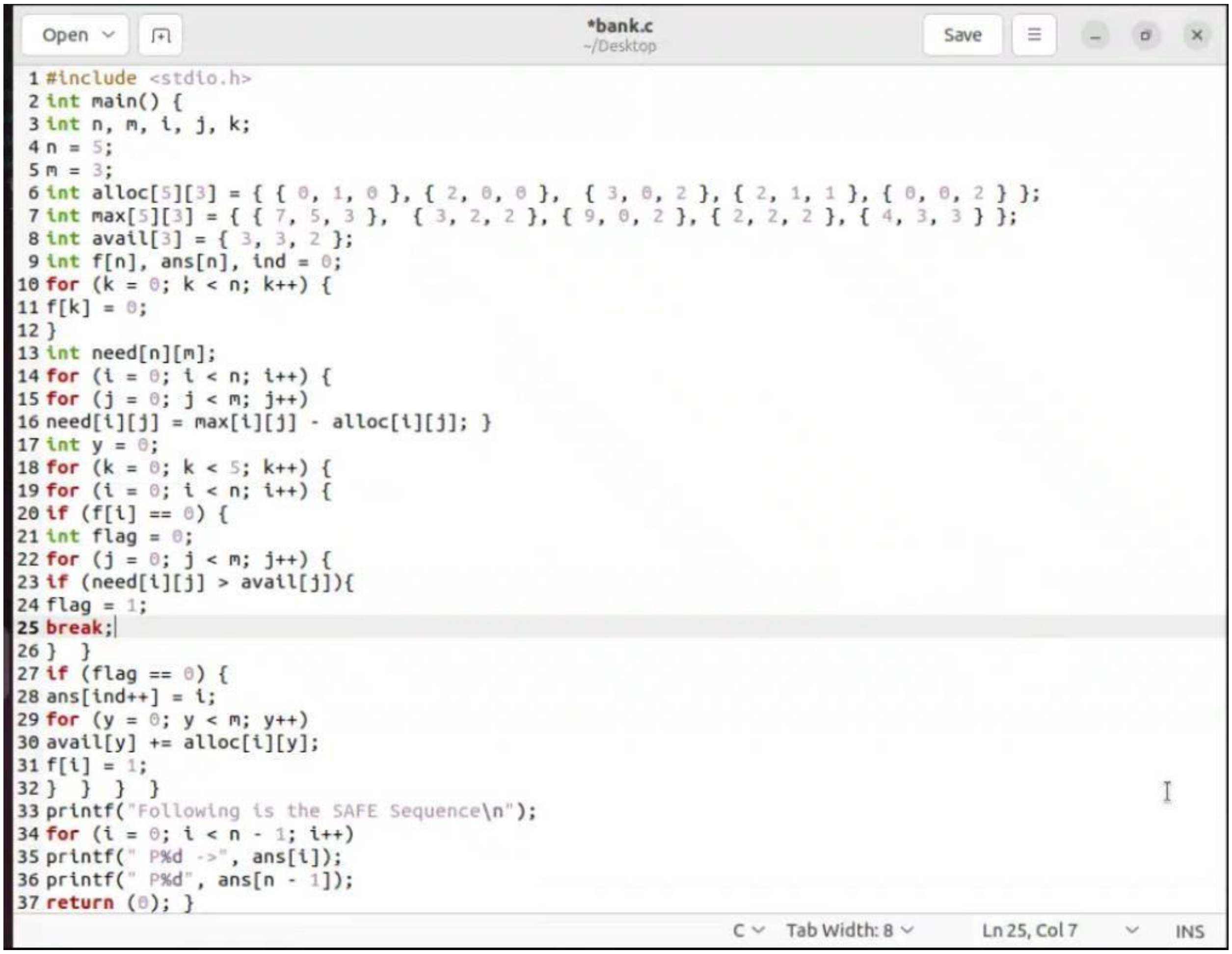
*********************** OUTPUT ***********************



```
onworks@onworks:~/Desktop$ touch new.c
onworks@onworks:~/Desktop$ gcc new.c -o test5
onworks@onworks:~/Desktop$ ./test5
The child process terminated normally.onworks@onworks:~/Desktop$ 
```

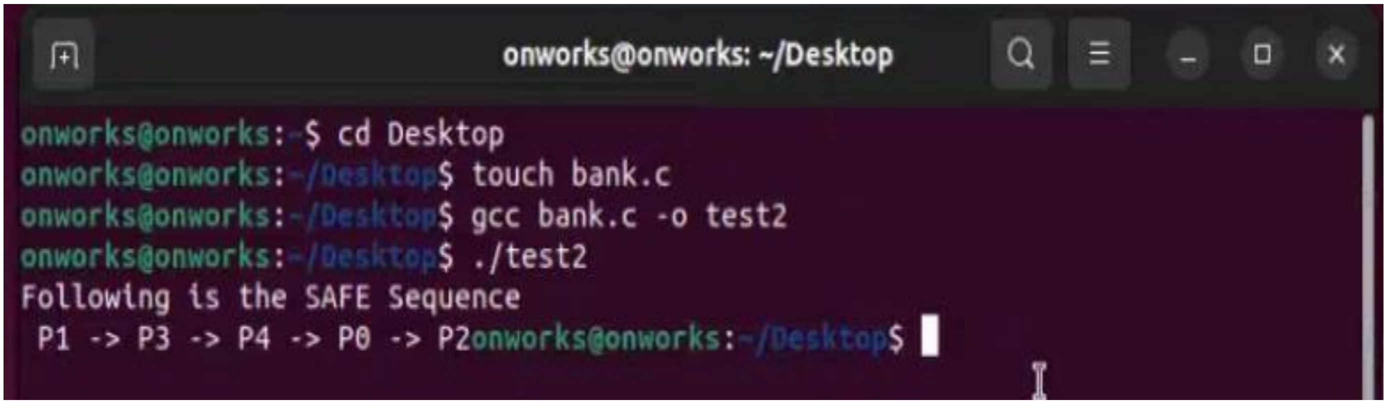


7. Write a program to demonstrate the concept of deadlock avoidance through bankers algorithm



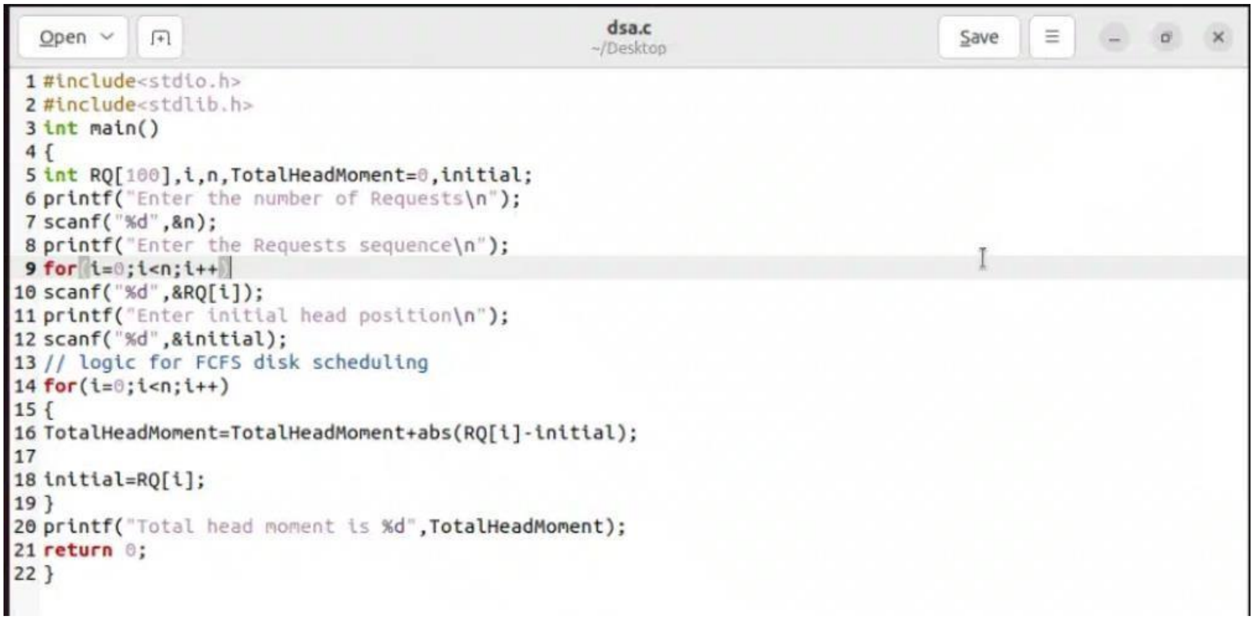
```c
#include <stdio.h>
int main() {
int n, m, i, j, k;
n = 5;
m = 3;
int alloc[5][3] = { { 0, 1, 0 }, { 2, 0, 0 },  { 3, 0, 2 }, { 2, 1, 1 }, { 0, 0, 2 } };
int max[5][3] = { { 7, 5, 3 },  { 3, 2, 2 }, { 9, 0, 2 }, { 2, 2, 2 }, { 4, 3, 3 } };
int avail[3] = { 3, 3, 2 };
int f[n], ans[n], ind = 0;
for (k = 0; k < n; k++) {
f[k] = 0;
}
int need[n][m];
for (i = 0; i < n; i++) {
for (j = 0; j < m; j++)
need[i][j] = max[i][j] - alloc[i][j]; }
int y = 0;
for (k = 0; k < 5; k++) {
for (i = 0; i < n; i++) {
if (f[i] == 0) {
int flag = 0;
for (j = 0; j < m; j++) {
if (need[i][j] > avail[j]){
flag = 1;
break;
}  }
if (flag == 0) {
ans[ind++] = i;
for (y = 0; y < m; y++)
avail[y] += alloc[i][y];
f[i] = 1;
}  }  }  }
printf("Following is the SAFE Sequence\n");
for (i = 0; i < n - 1; i++)
printf(" P%d ->", ans[i]);
printf(" P%d", ans[n - 1]);
return (0); }
```

******************** OUTPUT *******************



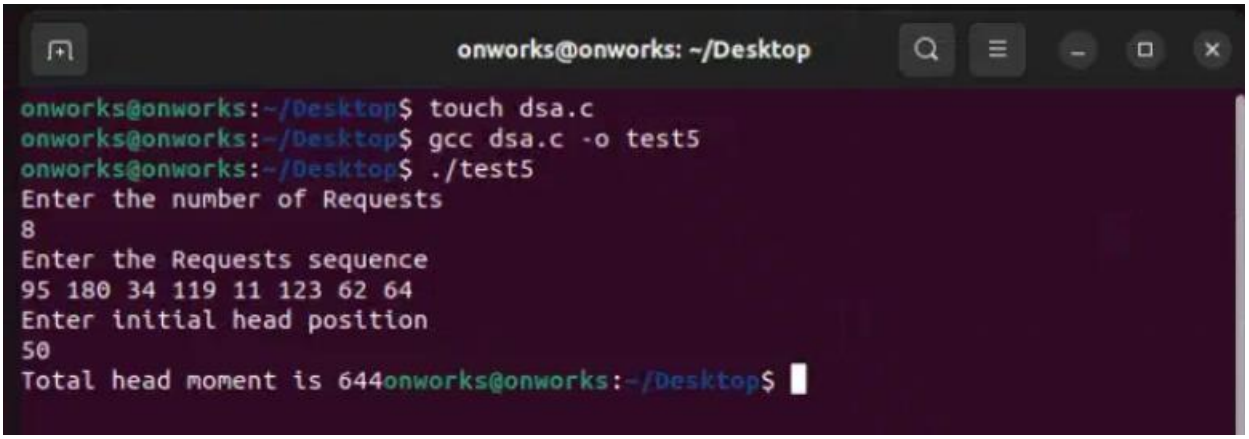
```
onworks@onworks:~$ cd Desktop
onworks@onworks:~/Desktop$ touch bank.c
onworks@onworks:~/Desktop$ gcc bank.c -o test2
onworks@onworks:~/Desktop$ ./test2
Following is the SAFE Sequence
 P1 -> P3 -> P4 -> P0 -> P2onworks@onworks:~/Desktop$ 
```

8. Write a program in c to do disk scheduling – FCFS



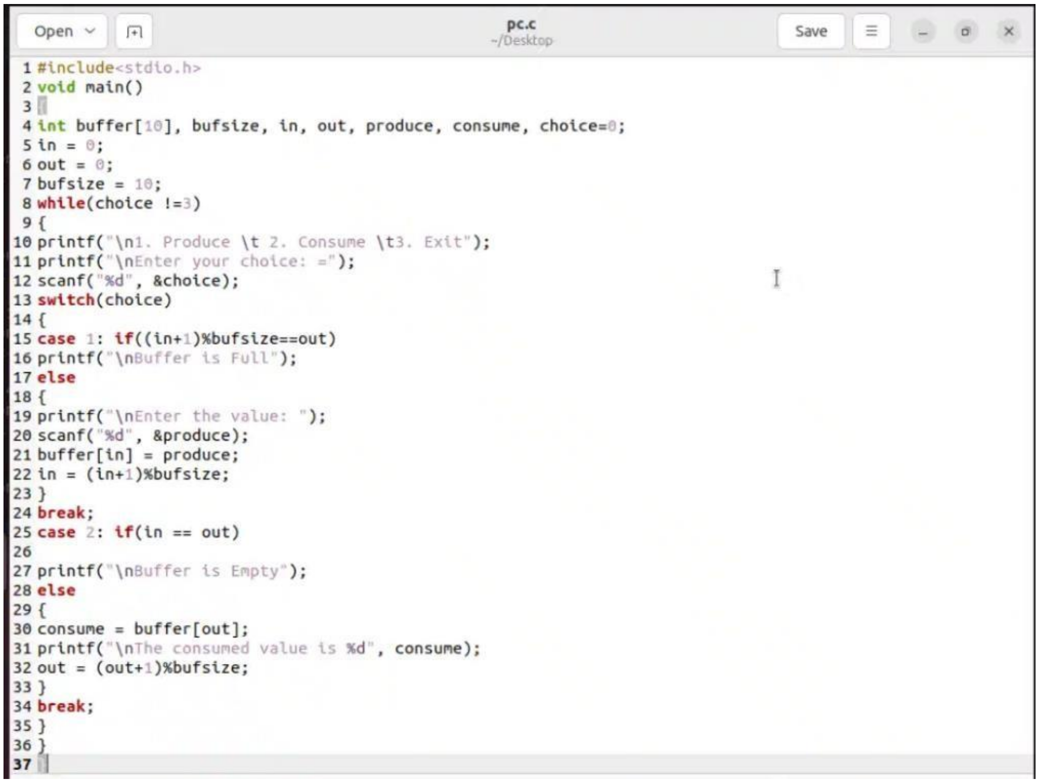
```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
int RQ[100],i,n,TotalHeadMoment=0,initial;
printf("Enter the number of Requests\n");
scanf("%d",&n);
printf("Enter the Requests sequence\n");
for(i=0;i<n;i++)
scanf("%d",&RQ[i]);
printf("Enter initial head position\n");
scanf("%d",&initial);
// logic for FCFS disk scheduling
for(i=0;i<n;i++)
{
TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);

initial=RQ[i];
}
printf("Total head moment is %d",TotalHeadMoment);
return 0;
}
```

******************** OUTPUT ***********************



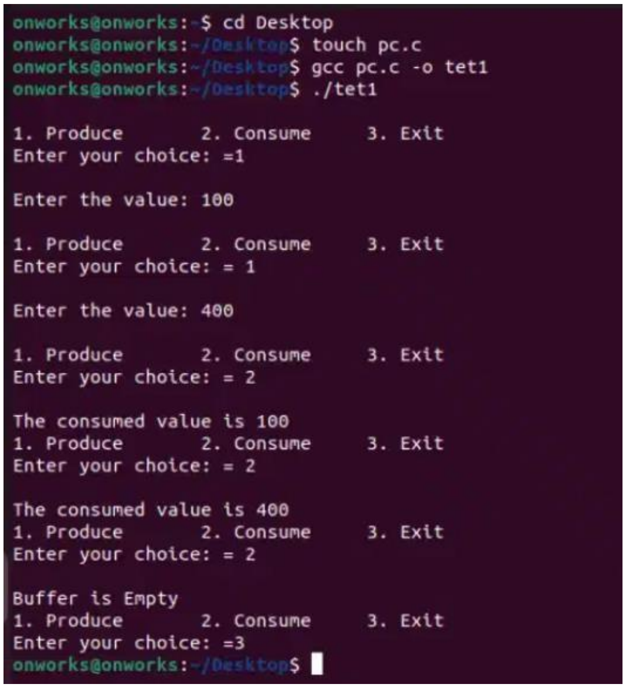
```
onworks@onworks:~/Desktop$ touch dsa.c
onworks@onworks:~/Desktop$ gcc dsa.c -o test5
onworks@onworks:~/Desktop$ ./test5
Enter the number of Requests
8
Enter the Requests sequence
95 180 34 119 11 123 62 64
Enter initial head position
50
Total head moment is 644onworks@onworks:~/Desktop$
```

9. Write a C program to implement solution of Producer consumer problem through Semaphore.



```c
#include<stdio.h>
void main()
{
int buffer[10], bufsize, in, out, produce, consume, choice=0;
in = 0;
out = 0;
bufsize = 10;
while(choice !=3)
{
printf("\n1. Produce \t 2. Consume \t3. Exit");
printf("\nEnter your choice: =");
scanf("%d", &choice);
switch(choice)
{
case 1: if((in+1)%bufsize==out)
printf("\nBuffer is Full");
else
{
printf("\nEnter the value: ");
scanf("%d", &produce);
buffer[in] = produce;
in = (in+1)%bufsize;
}
break;
case 2: if(in == out)

printf("\nBuffer is Empty");
else
{
consume = buffer[out];
printf("\nThe consumed value is %d", consume);
out = (out+1)%bufsize;
}
break;
}
}

```

************************ OUTPUT *********************



```
onworks@onworks:~$ cd Desktop
onworks@onworks:~/Desktop$ touch pc.c
onworks@onworks:~/Desktop$ gcc pc.c -o tet1
onworks@onworks:~/Desktop$ ./tet1

1. Produce       2. Consume     3. Exit
Enter your choice: =1

Enter the value: 100

1. Produce       2. Consume     3. Exit
Enter your choice: = 1

Enter the value: 400

1. Produce       2. Consume     3. Exit
Enter your choice: = 2

The consumed value is 100
1. Produce       2. Consume     3. Exit
Enter your choice: = 2

The consumed value is 400
1. Produce       2. Consume     3. Exit
Enter your choice: = 2

Buffer is Empty
1. Produce       2. Consume     3. Exit
Enter your choice: =3
onworks@onworks:~/Desktop$
```

5)

```
#include<stdio.h>
 #include<stdlib.h>
#include<sys/types.h>
 #include<sys/wait.h>
#include<unistd.h>
 int main(void) { pid_t pid = fork();
 if(pid==0) {
 printf( Child=>PPID:%d PID:%d\n",getppid(),getpid());
 exit(EXIT_SUCCESS);
}
else if(pid>0) {
 printf( “Paretnt=>PID:%d\n",getpid());
 printf("Waiting for child process to finish.\n”);
 wait(NULL);
 printf("Child process finished\n");}
 else{
printf("Unable to create child process.\n");
}
 return EXIT SUCCESS;
}
```

6)

```
 #include<stdio.h>
#include<stdlib.h>
 #include<sys/types.h>
 #include<sys/wait.h>
#include<unistd.h>
 int main(void) {
 pid_t childpid;
 childpid = fork();
if(childpid==0){
 exit(0);  }
 else if(childpid < 0) {
//log the error
printf(“Log the error\n");
}
 else{
 int returnStatus;
 waitpid(childpid, &returnStatus, 0);

if(returnStatus == 0)
 {
 printf( “The child process terminated normally.”);
}
 if(returnStatus == 1)
 {
 printf("The child process terminated with an errort”);
}
}}
```

7) Program

```
#include<stdio.h>
 #include<stdlib.h>
#include<sys/types.h>
 #include<sys/walt.h>
#include<unistd.h>
 int main(void) { pid_t pid = fork();
 if(pid==0) {
 printf( %d PID:%d\n",getppid().getpid());
 exit(EXIT_SUCCESS);
else if(ptd>0) {
 printf( PID:%d\n",getpid());
 printf("Waiting for child process to finish.in);
 wait(NULL);
 printf("Child process finished\n");
 else{
printf("Unable to create child process.\n");
}
 return EXIT SUCCESS;}
```

8) Program

```
#include<stdio.h>
 #include<stdlib.h>
#include<sys/types.h>
 #include<sys/walt.h>
#include<unistd.h>
 int main(void) { pid_t pid = fork();
 if(pid==0) {
 printf( %d PID:%d\n",getppid().getpid());
 exit(EXIT_SUCCESS);
else if(ptd>0) {
 printf( PID:%d\n",getpid());
 printf("Waiting for child process to finish.in);
 wait(NULL);
 printf("Child process finished\n");
 else{
printf("Unable to create child process.\n");
}
 return EXIT SUCCESS;}
```

9) Program

```
#include<stdio.h>
 #include<stdlib.h>
#include<sys/types.h>
 #include<sys/walt.h>
#include<unistd.h>
 int main(void) { pid_t pid = fork();
 if(pid==0) {
 printf( %d PID:%d\n",getppid().getpid());
 exit(EXIT_SUCCESS);
else if(ptd>0) {
 printf( PID:%d\n",getpid());
 printf("Waiting for child process to finish.in);
 wait(NULL);
 printf("Child process finished\n");
 else{
printf("Unable to create child process.\n");
}
 return EXIT SUCCESS;}
```