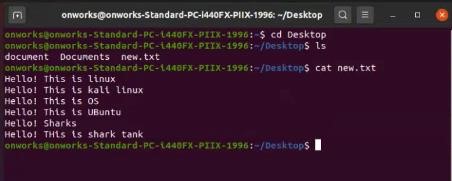
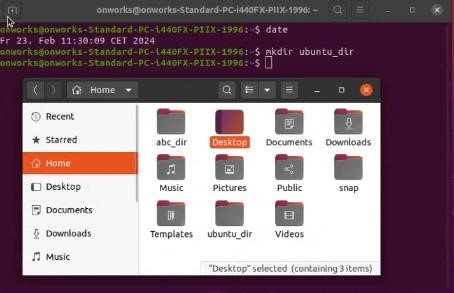
**1. Explore basic LINUX commands like mkdir, chdir, cat, ls,chmod.**

* 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.





**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

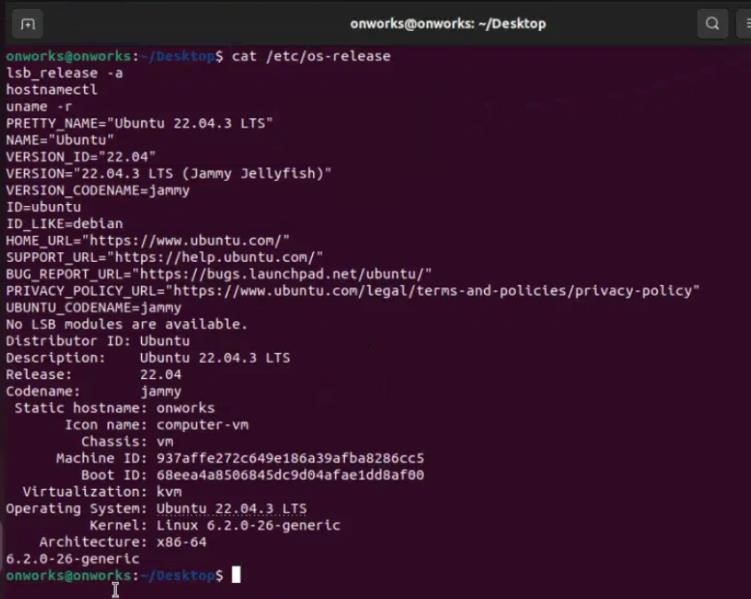
#!/bin/sh

$ps –eo size,command –sort –size |head

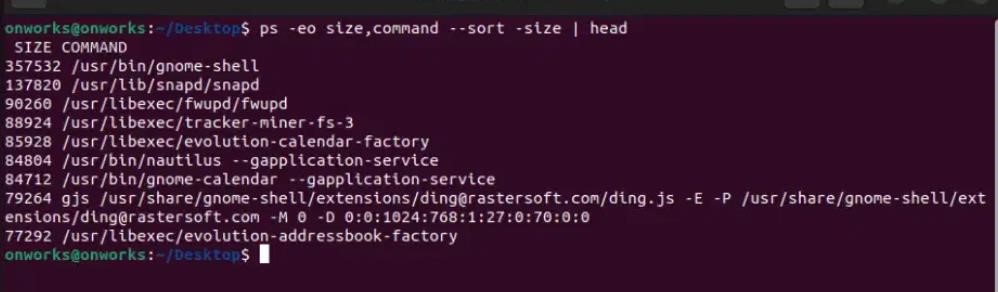
1. Program for shell script



1. Display os version,Release Number, Kernel Version



3.Display Top 10 processes in descending order



**4. Write shell scripts to do the following:**

a. Display processes with highest memory usage.

#!/bin/sh

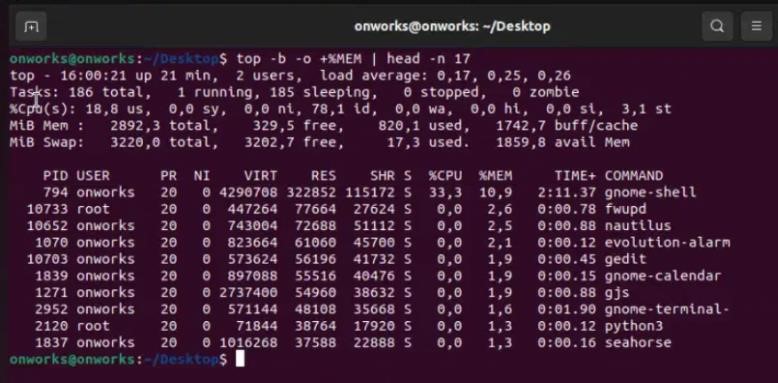
Top –b –o +%MEM | head -17

b.Display current logged in user and log name.

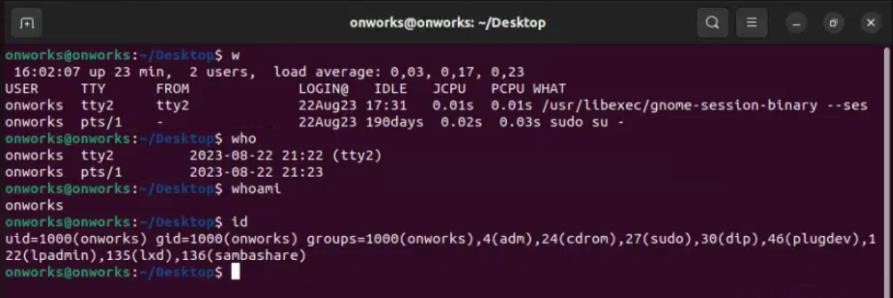
1. Program for shell script



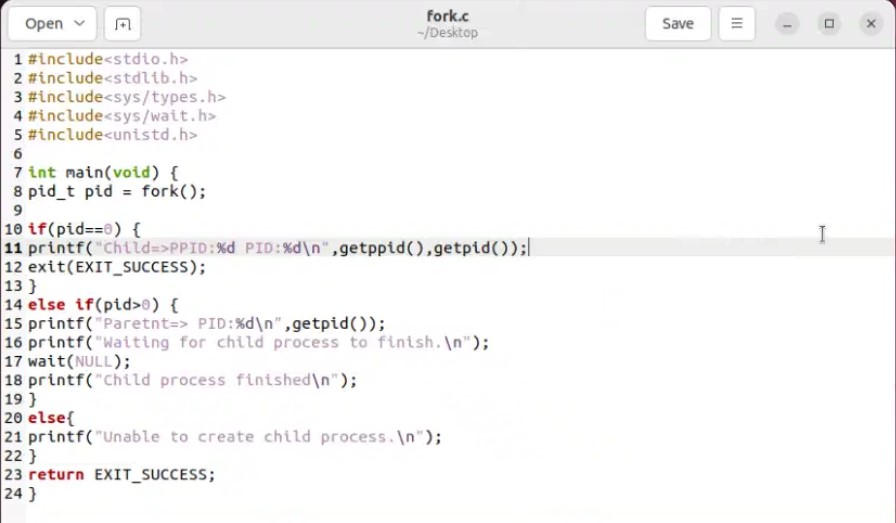
2.Display process with highest memory uses



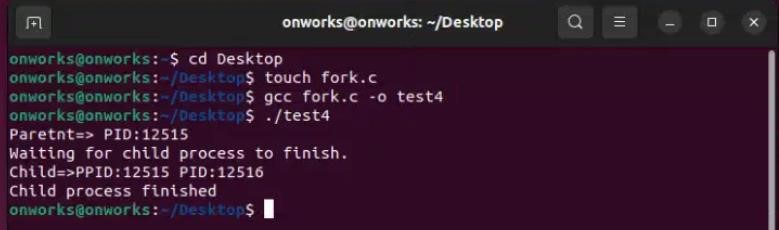
3. Display current logged in user and log name



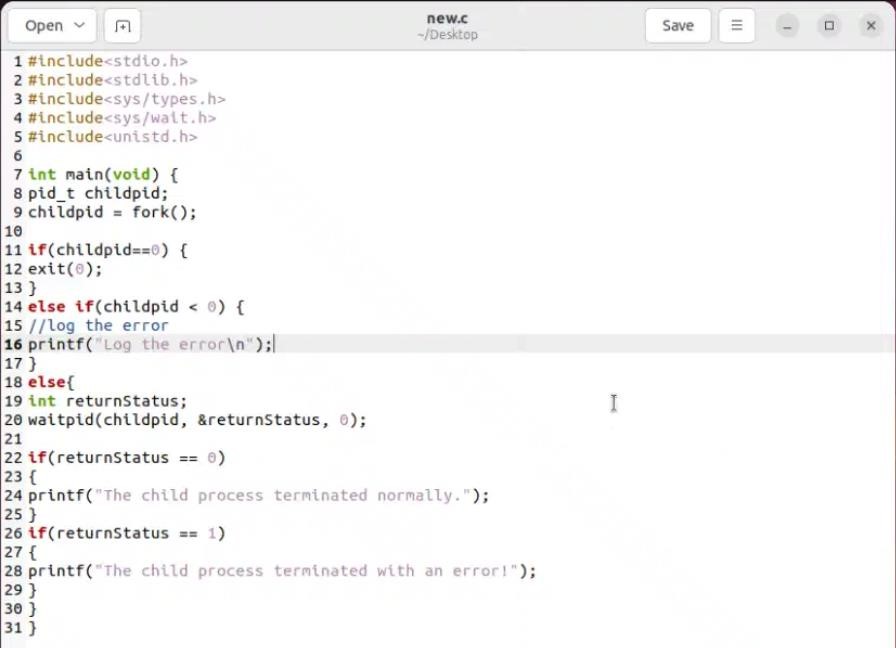
**5. Create a child process in Linux using the fork system call. From the child process obtain the process ID of both child and parent by using getpid and getppid system call.**



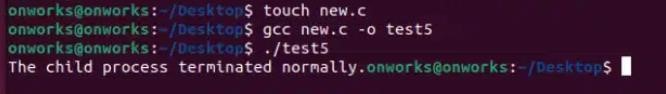
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* OUTPUT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



**6. Explore wait and waitpid before termination of process.**

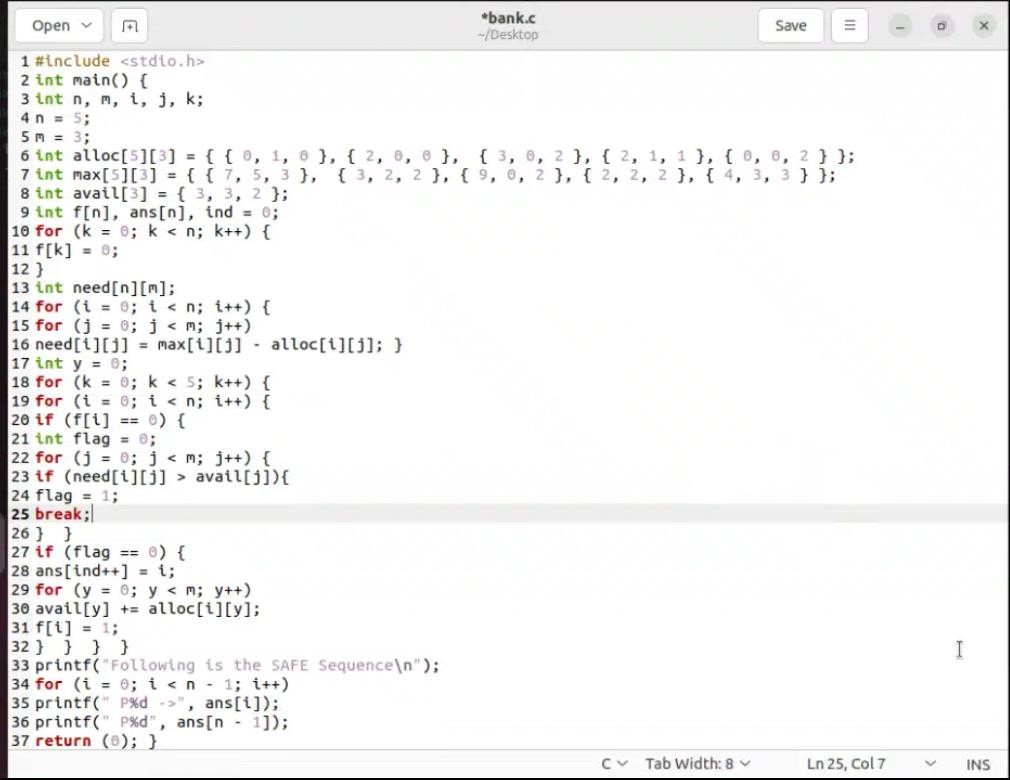


# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* OUTPUT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

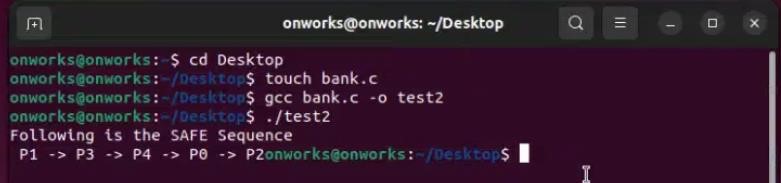




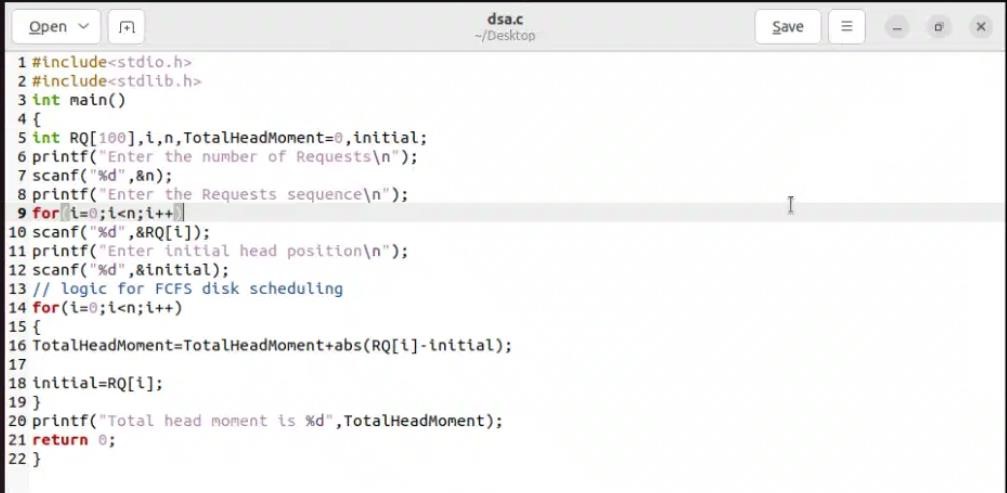
**7. Write a program to demonstrate the concept of deadlock avoidance through bankers algorithm**



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* OUTPUT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



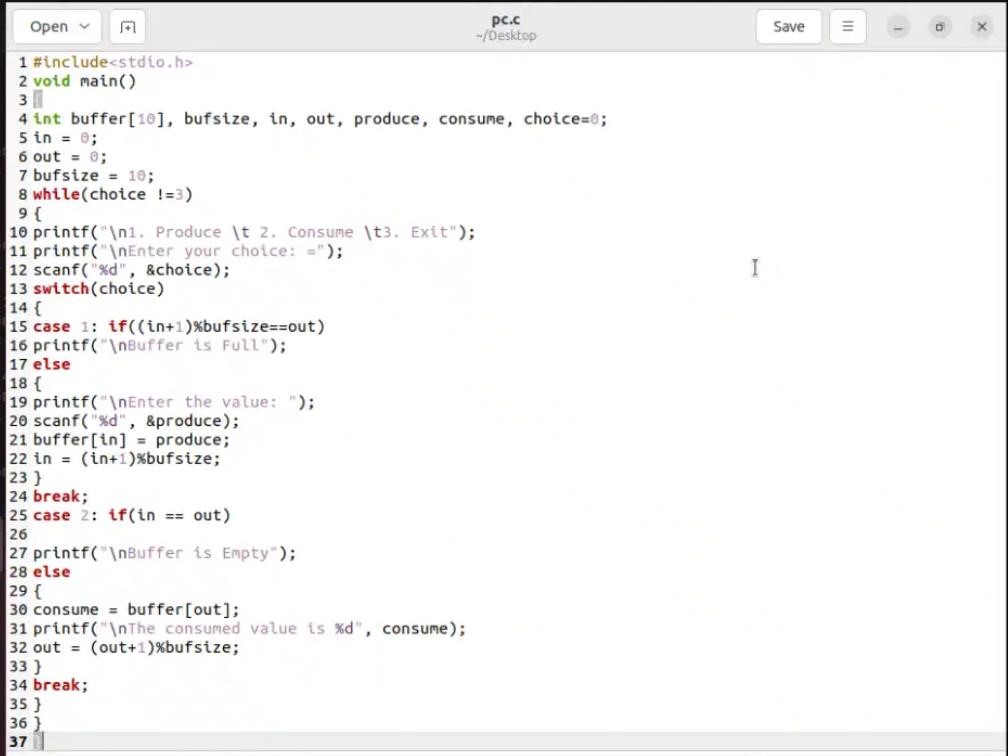
**8. Write a program in c to do disk scheduling – FCFS**



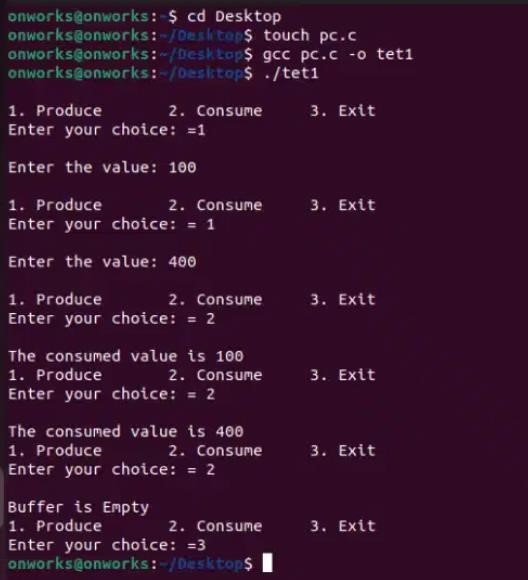
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* OUTPUT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



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



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* OUTPUT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



5)

#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;}

6)

1 #include<stdio.h>

2 #include<stdlib.h>

3 #include<sys/types.h>

4 #include<sys/wait.h>

5 #include<unistd.h>

6

7 int main(void) {

8 pid\_t childpid;9 childpid = fork();

10

11 if(childpid==0){ 12 exit(0);

13}

14 else if(childpid < 0) {

15//log the error

16 printf(“Log the error\n"); 17}

18 else{

19 int returnStatus;

20 waitpid(childpid, &returnStatus, 0);

21

22 if(returnStatus == 0)

23. {

24 printf( “The child process terminated normally.”);

25}26 if(returnStatus == 1)

27 {28 printf("The child process terminated with an errort);

29}30}31}

1. Program

#include <stdio.h>

2 int main() {

3 int n, m, i, j, k;

4 n = 5;

5 m = 3;

6 int alloc[5][3]={{0,1,0},{2,0,0},{3,0,2},{2,1,1},{0,0,2}};

7 int max[5][3]={{7,5,3), (3, 2, 2), (9, 0, 2), (2,2,2), (4,3,3));

8 int avail[3]={(3,3,2);

9 int f[n], ans[n], ind=0;

10 for (k = 0; k < n; k++) {

11f [k]=0; 12 }

13 int need[n][m];

14 for (i=0; i < n; i++) {

15 for (j=0;j<m; j++) ;

16 need[i][j] = max[i][j] alloc[i][j]; }

17 int y=0;

18 for (k=0; k < 5; k++) {

19 for(i=0;I<n;;i++){

20 if (f[i]==0){

21 int flag=0;

22 for (j=0; j<m; j++) {

23 if (need[i][j] avall[j]){

24 flag1; 25 break;

26 }}

27 if (flag) { 28 ans [ind++]=i;

29 for (y=0; y < m; y++)

30 avail[y] + =alloc [i] [y]:

31 f[i]=1;

32 }}}}

33 printf("Following is the SAFE Sequence\n");

34 for (i=0; i<n-1; i++) 35 printf(“ P%d->”, ans[i]);

36 printf("p%d", ans [n- 1]);

37 return (); }

1. Program

#include<stdio.h>

#include<stdlib.h>

int main()

{

int RQ[100],i,n,TotalHeadmoment=0,inital;

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);

for(i=0;i<n;i++)

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);

initial=RQ[i];

}

printf("total head moment is %d",TotalHeadMoment);

return 0;

}

1. Program

#include<stdio.h>

2 void main()

3

4 int buffer[10], bufsize, in, out, produce, consume, choice=0;

5 int = 0;

6 out =0;

7 bufsize= 10;

8 while(chotce 1=3)

9{

10 printf("\n1. Produce \t 2. Consume \t3. Exit");

11 printf("\nEnter your choice: =");

12 scanf("%d", &choice);

13 switch(choice)

14 {

15 case 1: if((in+1)%bufsize==out)

16 printf("\nBuffer ts Full);

17 else 18 {

19 printf("\nEnter the value: ");

20 scanf("%d", &produce);

21 buffer[in] = produce;

22 in = (in+1)%bufsize; 23}

24 break;

25 case 2: if(in == out) 26

27 printf("\nBuffer is Empty");

28 else 29 {

30 consume = buffer [out];

31 printf("\nThe consumed value is %d", consume);

32 out (out+1)%bufsize;

33}

34 break;

35}}