OPERATING SYSTEM LAB

**A PRACTICAL FILE**

***Submitted in partial fulfilment of the requirements for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

***in***

**COMPUTER SCIENCE AND TECHNOLOGY**

***by***

**PRAKHAR MAMGAIN**

**( EN. NO. GE- 192015394 )**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GRAPHIC ERA DEEMED TO BE UNIVERSITY, DEHRADUN**

**DEHRADUN – 248002 (INDIA)**

**Novemeber, 2021**

**PROGRAM NO. 1**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Demonstration of FORK() System Call - Single FORK()**

**Solution:**

#include<stdio.h>

#include<unistd.h>

int main()

{

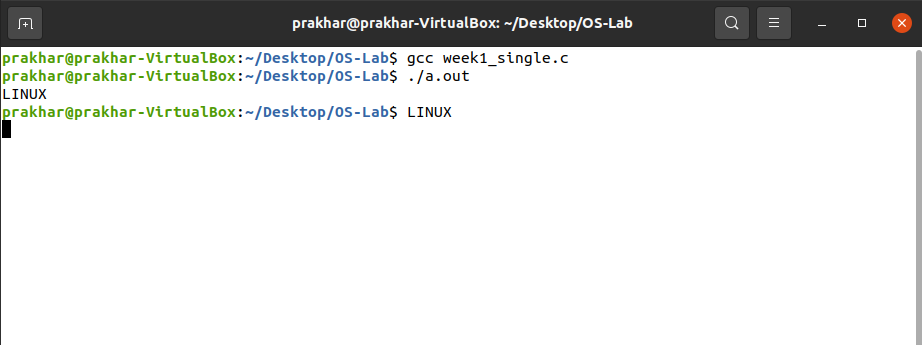
fork();

printf("LINUX\n");

return 0;

}

**Output:**



**PROGRAM NO. 1**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Demonstration of FORK() System Call - Multiple FORK()**

**Solution:**

#include<stdio.h>

#include<unistd.h>

int main()

{

fork();

printf("LINUX\n");

fork();

printf("UNIX\n");

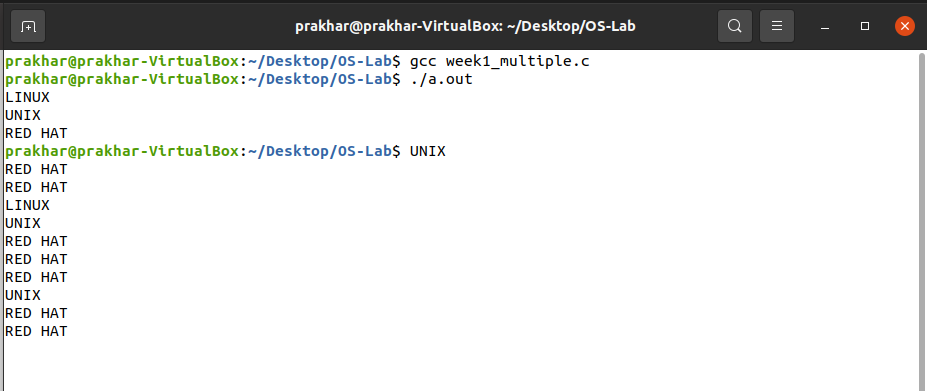
fork();

printf("RED HAT\n");

return 0;

}

**Output:**



**PROGRAM NO. 2**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Parent Process Computes the SUM OF EVEN and Child Process Computes the sum of ODD NUMBERS using fork()**

**Solution:**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/wait.h>

#define max 20

int main()

{

pid\_t pid;

int a[max],n,sum=0,i,status;

printf("Enter the no of terms in the array : ");

scanf("%d",&n);

printf("\nEnter values in the array : ");

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

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

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

pid=fork();

wait(&status);

if(pid == 0){

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

if(a[i]%2 == 0)

sum=sum+a[i];

printf("Sum of even nos = %d\n",sum);

exit(0);

}

else{

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

if(a[i]%2!=0)

sum=sum+a[i];

printf("Sum of odd nos = %d\n",sum);

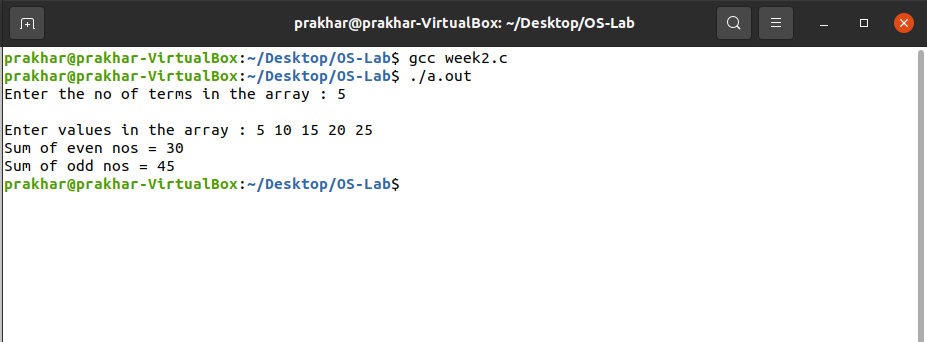
}

}

return 0;

}

**Output:**



**PROGRAM NO. 3**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Demonstration of WAIT() System Call**

**Solution:**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/wait.h>

int main()

{

pid\_t pid;

int status;

pid = fork();

if(pid == 0)

{

printf("I m Child\n");

exit(0);

}

else

{

wait(& status);

printf("I m Parent\n");

printf("The Child PID = %d\n",pid);

}

return 0;

}

**Output:**



**PROGRAM NO. 4**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Implementation of ORPHAN PROCESS**

**Solution:**

#include<stdio.h>

#include<unistd.h>

int main()

{

pid\_t pid;

pid = fork();

if(pid == 0){

sleep(6);

printf("I am CHILD.");

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

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

}

else if(pid>0){

printf("I am PARENT");

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

printf("\nGrand Parent Process: %d", getpid());

printf("\nTERMINATING PID = %d\n",getpid());

}

return 0;

}

**Output:**



**PROGRAM NO. 4**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Implementation of ZOMBIE PROCESS**

**Solution:**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

int main()

{

pid\_t pid = fork();

if(pid!=0){

while(1)

sleep(50);

}

else

exit(0);

return 0;

}

**Output:**

**PROGRAM NO. 5**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Implementation of PIPE**

**Solution:**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<string.h>

int main()

{

pid\_t pid;

char arr[100], str[100];

int fd[2],nbr,nbw;

pipe(fd);

pid = fork();

if(pid == 0){

printf("Enter a string : ");

gets(str);

nbw = write(fd[1],str,strlen(str));

printf("Child Wrote %d bytes\n ", nbw);

exit(0);

}

else{

nbr = read(fd[0], arr, sizeof(arr));

arr[nbr]='\0';

printf("Parent read %d bytes: %s",nbr,arr);

}

return 0;

}

Output:



**PROGRAM NO. 6**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Implementation of FIFO**

**Solution:**

1. **FIFO (WRITER PROCESS):**

#include<stdio.h>

#include<string.h>

#include<sys/stat.h>

#include<fcntl.h>

int main()

{

int fd, nbw;

char str[100];

mknod("myfifo",S\_IFIFO|0666,0);

printf("Writing for reader Process:\n\t");

fd=open("myfifo",O\_WRONLY);

while(gets(str, 100, stdin))

{

nbw=write(fd,str,strlen(str));

printf("Writer process write %d bytes: %s\n",nbw,str);

}

return 0;

}

1. **FIFO (READER PROCESS):**

#include<stdio.h>

#include<string.h>

#include<sys/stat.h>

#include<fcntl.h>

int main()

{

int fd ,nbr;

char arr[100];

mknod("myfifo",S\_IFIFO|0666,0);

fd=open("myfifo",O\_RDONLY);

printf("If you got a writer process then type some data \n");

do{

nbr=read(fd,arr,sizeof(arr)); arr[nbr]='\0';

printf("Reader process read %d bytes: %s\n",nbr,arr);

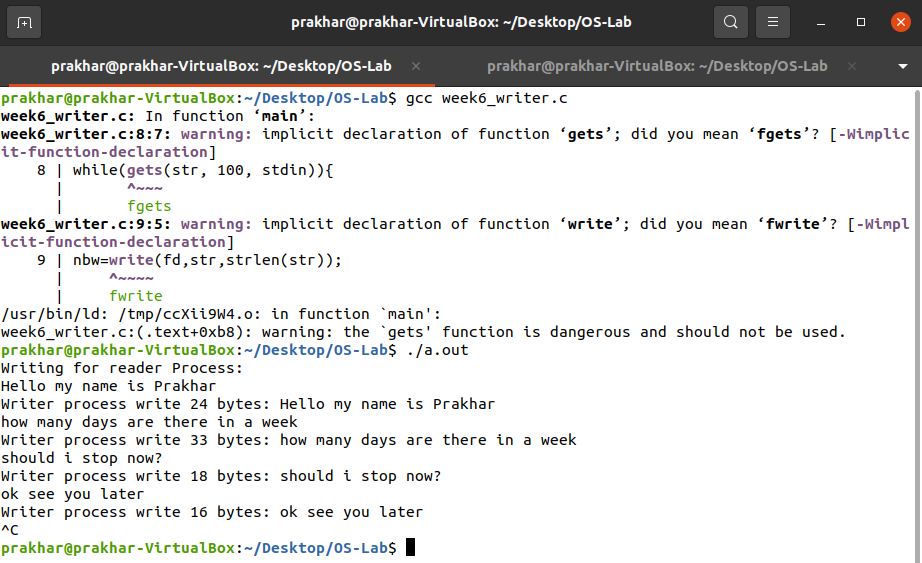
}while(nbr>0);

return 0;

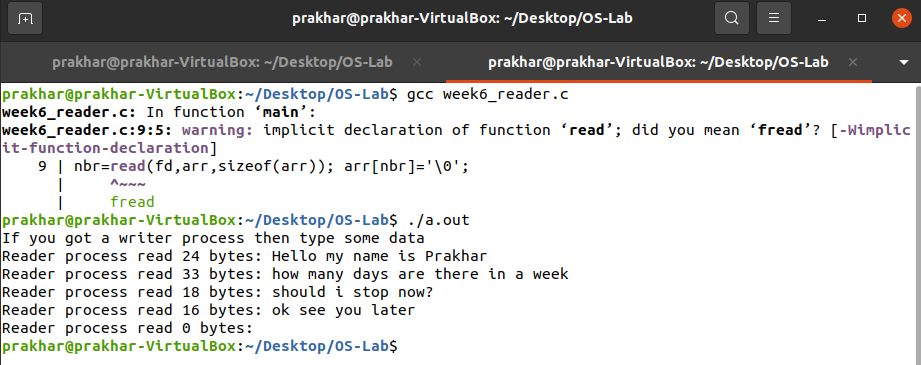
}

**Output:**

1. **FIFO (WRITER PROCESS):**



1. **FIFO (READER PROCESS):**



**PROGRAM NO. 7**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Implementation of MESSAGE QUEUE**

**Solution:**

1. **WRITER PROCESS:**

#include<stdio.h>

#include<string.h>

#include<sys/ipc.h>

#include<sys/msg.h>

#include<sys/types.h>

struct msgbuf{

long mtype;

char mtext[100];

}svarname;

int main(){

key\_t key;

int msgid ,c;

key=ftok("progfile",'A');

msgid=msgget(key,0666|IPC\_CREAT);

svarname.mtype=1;

printf("\nEnter a string : "); gets(svarname.mtext);

c=msgsnd(msgid,&svarname,strlen(svarname.mtext),0);

printf("Sender wrote the text :\t %s \n",svarname.mtext);

return(0);

}

1. **READER PROCESS:**

#include<stdio.h>

#include<string.h>

#include<sys/ipc.h>

#include<sys/msg.h>

#include<sys/types.h>

struct msgbuf{

long mtype;

char mtext[100];

}svarname;

int main(){

key\_t key;

int msgid ,c;

key=ftok("progfile",'A');

msgid=msgget(key,0666|IPC\_CREAT); msgrcv(msgid,&svarname,sizeof(svarname),1,0);

printf("Data Received is : \t %s \n",svarname.mtext);

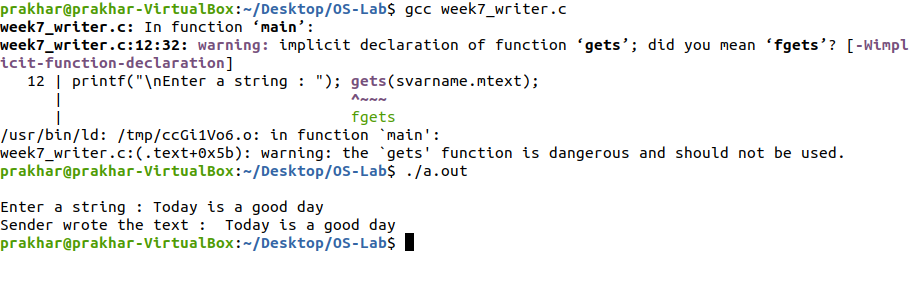
msgctl(msgid,IPC\_RMID,NULL);

return 0;

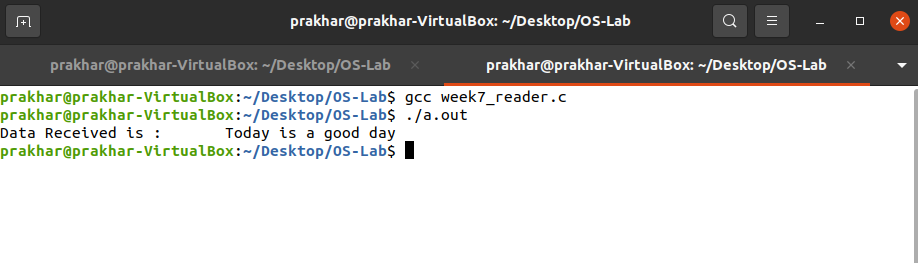
}

**Output:**

1. **WRITER PROCESS:**



1. **READER PROCESS:**



**PROGRAM NO. 8**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Implementation of SHARED MEMORY**

**Solution:**

1. **WRITER PROCESS:**

#include<stdio.h>

#include<string.h>

#include<sys/ipc.h>

#include<sys/shm.h>

#include<sys/types.h>

int main()

{

key\_t key;

int shmid;

void \*ptr;

key=ftok("shmfile",'A');

shmid=shmget(key,1024,0666|IPC\_CREAT);

ptr=shmat(shmid,(void \*)0,0); printf("\nInput Data: ");

gets(ptr);

shmdt(ptr);

return 0;

}

1. **READER PROCESS:**

#include<stdio.h>

#include<string.h>

#include<sys/ipc.h>

#include<sys/shm.h>

#include<sys/types.h>

int main()

{

key\_t key;

int shmid; v

oid \*ptr;

key=ftok("srfile",'A');

shmid=shmget(key,1024,0666|IPC\_CREAT);

ptr=shmat(shmid,(void \*)0,0);

printf("\nThe Data stored : %s\n",ptr);

shmdt(ptr); shmctl(shmid,IPC\_RMID,NULL);

return 0;

}

1. **SHARED MEMORY:**

#include<stdio.h>

#include<string.h>

#include<sys/ipc.h>

#include<sys/shm.h>

#include<sys/types.h>

int main()

{

key\_t key;

int shmid;

void \*ptr;

key=ftok("srfile",'A');

shmid=shmget(key,1024,0666|IPC\_CREAT);

ptr=shmat(shmid,(void \*)0,0);

printf("\nInput Data:"); gets(ptr);

printf("\nThe Data stored : %s\n",ptr);

shmdt(ptr);

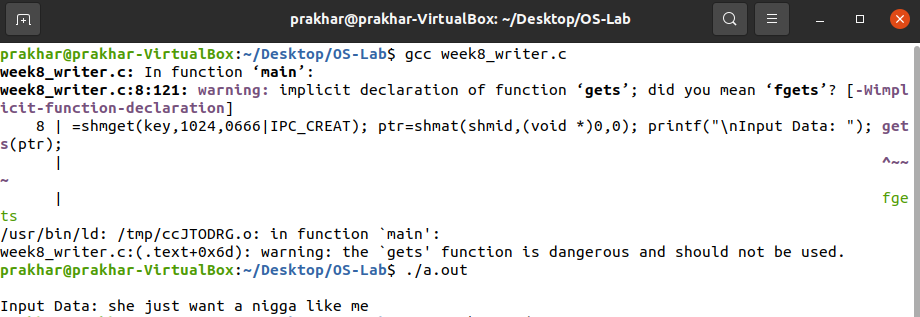
shmctl(shmid,IPC\_RMID,NULL);

return 0;

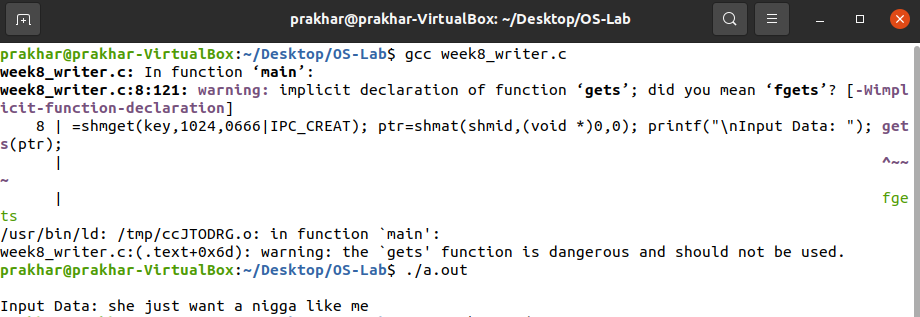
}

**Output:**

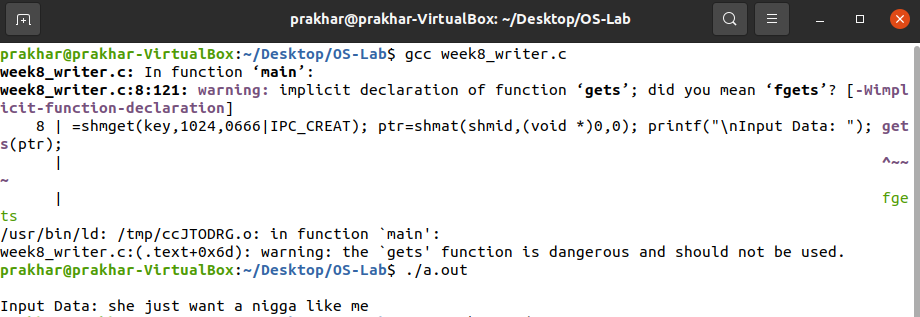
1. **WRITER PROCESS**



1. **READER PROCES**



1. **SHARED MEMORY**



**PROGRAM NO. 9**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Implementation of FIRST COME FIRST SERVED(FCFS) using arrays**

**Solution:**

#include<stdio.h>

int main(){

char p[10][5],temp[5];

int c=0,pt[10],i,j,n,temp1;

float bst=0.0,turn=0.0;

printf("enter no of processes:");

scanf("%d",&n);

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

printf("enter process%d name:\n",i+1);

scanf("%s",p[i]);

printf("enter process time");

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

}

printf("\n.....................................................\n");

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

printf("|\t %s\t",p[i]);

printf("|\n.....................................................\n");

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

printf("\t\t%d",pt[i]);

printf("\n.....................................................\n");

printf("0");

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

bst+=c;

turn+=c+pt[i];

c=c+pt[i];

printf("\t\t%d",c);

}

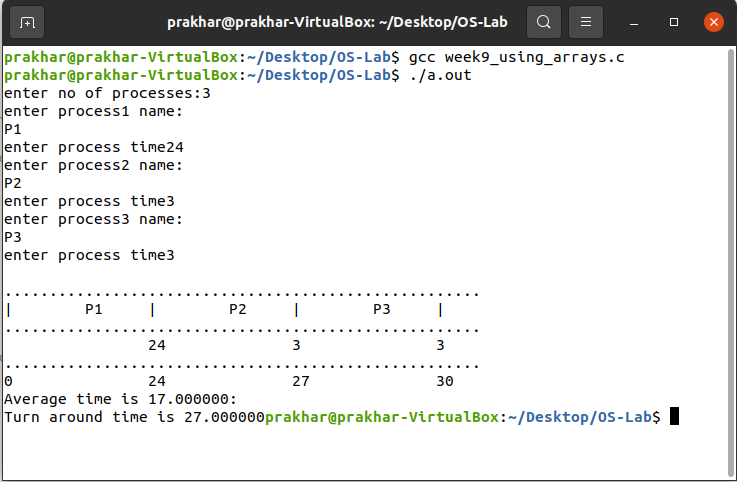
printf("\nAverage time is %f: ",bst/n);

printf("\nTurn around time is %f", turn/n);

return 0;

}

**Output:**



**PROGRAM NO. 9**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Implementation of FIRST COME FIRST SERVED(FCFS) using pointers**

**Solution:**

#include<stdio.h>

#include<malloc.h>

#include<string.h>

typedef struct node{

char prss[3];

int burst;

int arrival;

struct node \*next;

}node;

node \*front=NULL;

node \*rear=NULL;

void insert();

void display(int);

void main(){

int i ,n;

printf("\nEnter number of processes : ");

scanf("%d",&n);

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

insert();

printf("\n\nExecuting processes : \n");

display(n);

printf("\n");

return 0;

}

void insert(){

node \*p;

int b ,a;

char str[3];

p=(node\*)malloc(sizeof(node));

printf("\n\tEnter the process name : ");

scanf("%s",p->prss);

printf("\tEnter Burst time : ");

scanf("%d",&b);

printf("\tEnter arrival time : ");

scanf("%d",&a);

p->burst=b;

p->arrival=a;

p->next=NULL;

if(front==NULL){

front=p;

rear=p;

}

else{

rear->next=p;

rear=p;

}

}

void display(int n){

node \*temp=front;

int wttime=0,c=0;

float turn=0.0;

if(front!=NULL){

printf("\n---------------------------------------------------------------\n\t");

while(temp!=NULL){

printf("|\t%s\t",temp->prss);

temp=temp->next;

}

printf("|\n---------------------------------------------------------------\n\t");

temp=front;

while(temp!=NULL){

printf(" \t%d\t ",temp->burst);

temp=temp->next;

}

printf("\n-----------------------------------------------------------------\n\t");

temp=front;

printf("0\t");

while(temp!=NULL){

wttime+=c;

turn+=c+temp->burst;

c=c+temp->burst;

printf(" \t%d\t ",c);

temp=temp->next;

}

printf("\n-----------------------------------------------------------------\n");

printf("\n\nAveragewt time = %d ",wttime/n);

printf("\nTurnaround time = %f\n",turn/n);

}

}

**Output:**



**PROGRAM NO. 10**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Implementation of SHORTEST-JOB-FIRST-SCHEDULING(SJFS) using arrays**

**Solution:**

#include<stdio.h>

#include<string.h>

void main(){

char p[10][5],temp[5];

int c=0,pt[10],i,j,n,temp1;

float bst=0.0,turn=0.0;

printf("enter no of processes:");

scanf("%d",&n);

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

printf("enter process%d name:\n",i+1);

scanf("%s",p[i]);

printf("enter process time");

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

}

for(i=0;i<n-1;i++){

for(j=i+1;j<n;j++){

if(pt[i]>pt[j]){

temp1=pt[i];

pt[i]=pt[j];

pt[j]=temp1;

strcpy(temp,p[i]);

strcpy(p[i],p[j]);

strcpy(p[j],temp);

}

}

}

printf("\n.....................................................\n");

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

printf("|\t %s\t",p[i]);

printf("|\n.....................................................\n");

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

printf("\t\t%d",pt[i]);

printf("\n.....................................................\n");

printf("0");

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

bst+=c;

turn+=c+pt[i];

c=c+pt[i];

printf("\t\t%d",c);

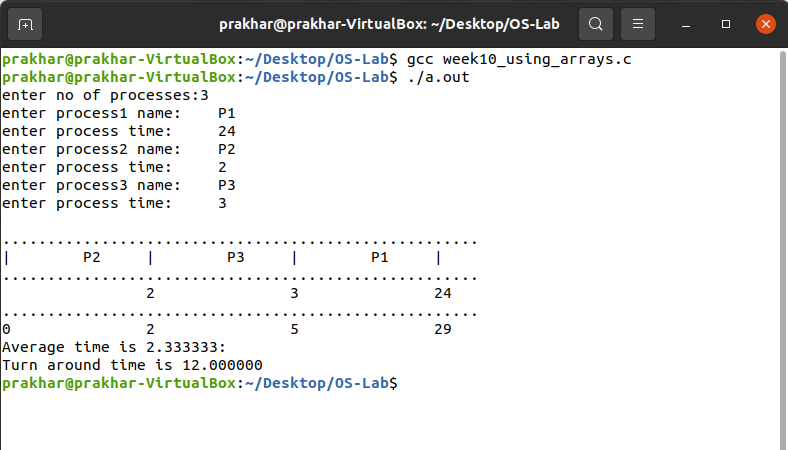
}

printf("\nAverage time is %f: ",bst/n);

printf("\nTurn around time is %f",turn/n);

}

**Output:**



**PROGRAM NO. 10**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Implementation of SHORTEST-JOB-FIRST-SCHEDULING(SJFS) using pointers**

**Solution:**

#include<stdio.h>

#include<malloc.h>

#include<string.h>

typedef struct node{

char prss[3];

int burst;

struct node \*next;

}node;

node \*front=NULL;

node \*rear=NULL;

void insert();

void display(int);

void main(){

int i,n;

printf("\nEnter number of processes : ");

scanf("%d",&n);

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

insert();

printf("\n\nExecuting processes : \n");

display(n);

printf("\n");

}

void insert(){

node \*p,\*temp;

int b;

p=(node\*)malloc(sizeof(node));

printf("\n\tEnter the process name : ");

scanf("%s",p->prss);

printf("\tEnter Burst time : ");

scanf("%d",&b);

p->burst=b;

p->next=NULL;

if(front==NULL){

front=p;

rear=p;

}

else if( p->burst < front->burst){

p->next=front;

front=p;

}

else if( p->burst > rear->burst){

rear->next=p;

rear=p;

}

else{

temp=front;

while( p->burst > (temp->next)->burst )

temp=temp->next;

p->next=temp->next;

temp->next=p;

}

}

void display(int n){

node \*temp=front;

int c=0;

float turn=0.0,wttime=0.0;

if(front!=NULL){

printf("\n---------------------------------------------------\n\t");

while(temp!=NULL){

printf("|\t%s\t",temp->prss);

temp=temp->next;

}

printf("\n-----------------------------------------------------------------\n\t");

temp=front;

while(temp!=NULL){

printf(" \t%d\t ",temp->burst);

temp=temp->next;

}

printf("\n----------------------------------------------------------------\n\t");

temp=front;

printf("0\t");

while(temp!=NULL){

wttime+=c;

turn+=c+temp->burst;

c=c+temp->burst;

printf(" \t%d\t ",c);

temp=temp->next;

}

printf("\n---------------------------------------------------------------------\n");

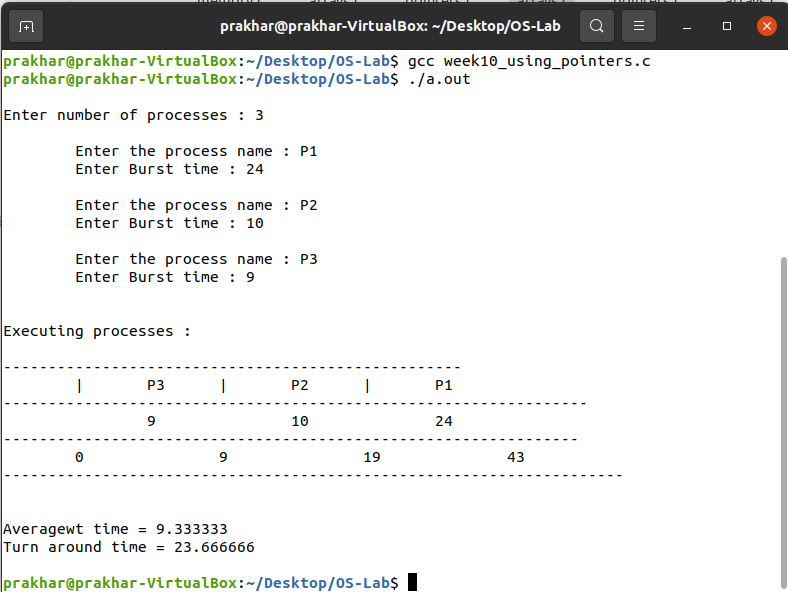
printf("\n\nAveragewt time = %f ",wttime/n);

printf("\nTurn around time = %f\n",turn/n);

}

}

**Output:**



**PROGRAM NO. 11**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Implementation of PRIORITY SCHEDULING using arrays**

**Solution:**

#include<stdio.h>

#include<string.h>

void main(){

char p[10][5],temp[5];

int c=0,pt[10],pr[10],i,j,n,temp1,t;

float bst=0.0,turn=0.0;

printf("enter no of processes:");

scanf("%d",&n);

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

printf("enter process%d name:\n",i+1);

scanf("%s",p[i]);

printf("enter process time");

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

printf("\nenter the priority of process");

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

}

for(i=0;i<n-1;i++){

for(j=i+1;j<n;j++){

if(pr[i]>pr[j]){

temp1=pt[i];

pt[i]=pt[j];

pt[j]=temp1;

t=pr[i];

pr[i]=pr[j];

pr[j]=t;

strcpy(temp,p[i]);

strcpy(p[i],p[j]);

strcpy(p[j],temp);

}

}

}

printf("\n.....................................................\n");

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

printf("|\t %s\t",p[i]);

printf("|\n.....................................................\n");

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

printf("\t\t%d",pt[i]);

printf("\n.....................................................\n");

printf("0");

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

bst+=c;

turn+=c+pt[i];

c=c+pt[i];

printf("\t\t%d",c);

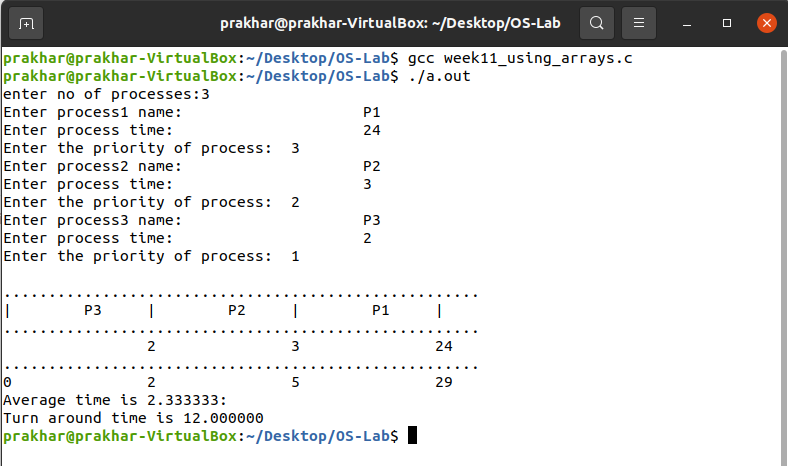
}

printf("\nAverage time is %f: ",bst/n);

printf("\nTurn around time is %f", turn/n);

}

**Output:**



**PROGRAM NO. 11**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Implementation of PRIORITY SCHEDULING using pointers**

**Solution:**

#include<stdio.h>

#include<malloc.h>

#include<string.h>

typedef struct node{

char prss[3];

int burst;

int priority;

struct node \*next;

}node;

node \*front=NULL;

node \*rear=NULL;

void insert();

void display(int);

void main(){

int i,n;

printf("\nEnter number of processes : ");

scanf("%d",&n);

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

insert();

printf("\n\nExecuting processes : \n");

display(n);

printf("\n");

}

void insert(){

node \*p,\*temp;

int b,pri;

p=(node\*)malloc(sizeof(node));

printf("\n\tEnter the process name : "); scanf("%s",p->prss);

printf("\tEnter Burst time : "); scanf("%d",&b);

printf("\tEnter Priority : "); scanf("%d",&pri);

p->burst=b;

p->priority=pri;

p->next=NULL;

if(front==NULL){

front=p;

rear=p;

}

else if(p->priority < front->priority){

p->next=front;

front=p;

}

else if(p->priority > rear->priority){

rear->next=p;

rear=p;

}

else{

temp=front;

while( p->priority > (temp->next)->priority )

temp=temp->next;

p->next=temp->next;

temp->next=p;

}

}

void display(int n){

node \*temp=front;

int c=0;

float turn=0.0,wttime=0.0;

if(front!=NULL){

printf("\n-------------------------------------------------------\n\t");

while(temp!=NULL){

printf("|\t%s\t",temp->prss);

temp=temp->next;

}

printf("\n-------------------------------------------------------------------\n");

temp=front;

while(temp!=NULL){

printf("\t%d\t ",temp->burst);

temp=temp->next;

}

printf("\n------------------------------------------------------------------\n\t");

temp=front;

printf("0\t");

while(temp!=NULL){

wttime+=c;

turn+=c+temp->burst;

c=c+temp->burst;

printf(" \t%d\t ",c);

temp=temp->next;

}

printf("\n------------------------------------------------------------\n");

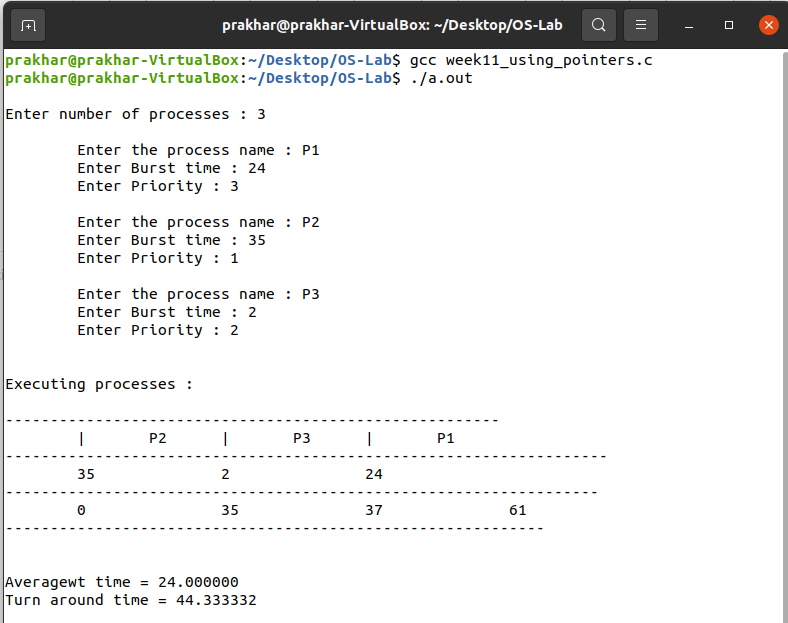
printf("\n\nAveragewt time = %f ",wttime/n);

printf("\nTurn around time = %f\n",turn/n);

}

}

**Output:**



**PROGRAM NO. 12**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Implementation of FIFO PAGE REPLACEMENT**

**Solution:**

#include<stdio.h>

#include<stdlib.h>

#define MAX 5

int front=0,back=-1,cs=0,nf;

int f[MAX];

void enq(int x);

void deq(void);

void dis(void);

int isfound(int);

void main(){

int pf=0,rfs,rf[15],i;

printf("\n FIFO page replacement");

printf("\n Enter the size of reference string:");

scanf("%d",&rfs);

printf("\n Enter the reference string:");

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

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

printf("\n Enter the number of free frames:");

scanf("%d",&nf);

enq(rf[0]);

pf=1;

for(i=0;i<rfs;i++){

if(!isfound(rf[i])){

pf++;

if(cs==nf)

deq();

enq(rf[i]);

}

dis();

}

printf("\n No of page faults :%d",pf);

}

int isfound(int x){

int i;

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

if(f[i]==x)

return 1;

return 0;

}

void enq(int x){

if(++back==nf)

back=0;

f[back]=x;

cs++;

}

void dis(){

int i;

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

printf("%d",f[i]);

printf("\n");

}

void deq(){

nf--;

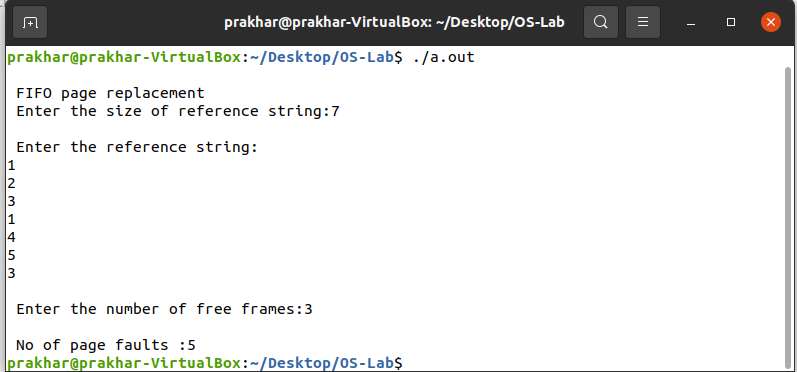
if(++front==nf)

front=0;

return;

}

Output:



**PROGRAM NO. 13**

**Name: Prakhar Mamgain**

**Roll No.: 35**

**Section: CST**

**Semester: 5th**

**Objective: Implementation of PAGE REPLACEMENT LRU**

**Solution:**

#include<stdio.h>

#include<stdlib.h>

int fsize,ssize,f,frame[10],arrive[30],rstring[30];

int pagefound(int x);

void display();

int leastused();

int pagelocation(int x);

int main(){

int i,lfi,idx,cs=0,f,ls=0,pf=0,j=0,y,k,z=0,time=0;

printf("\n\n\t\t LRU PAGE REPLACEMENT");

printf("\n\t\t --------------------");

printf("\n\n\t Enter the frame size:");

scanf("%d",&fsize);

printf("\n\t Enter the reference string size:");

scanf("%d",&ssize);

printf("\n\t Enter the reference string:");

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

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

for(k=0;k<fsize;k++){

frame[k]=-3;

arrive[k]=0;

}

for(i=0;i<ssize;i++){

y=pagefound(rstring[i]);

if(y==0){

pf++;

if(cs>=fsize){

lfi=leastused();

frame [lfi]=rstring[i];

arrive [lfi]=++time;

}

else if (cs<fsize){

frame[cs]=rstring[i];

arrive [cs]=++time;

}

}

else{

idx=pagelocation(rstring[i]);

arrive [idx]=++time;

}

cs++;

display();

}

printf("\n Page fault=%d",pf);

}

int pagefound(int x){

int i,val=0;

for(i=0;i<fsize;i++){

if(x==frame[i]){

val=1;

break;

}

}

return val;

}

void display(){

int i;

printf("\n");

for(i=0;i<fsize;i++){

if(frame[i]>=0)

printf("%d",frame[i]);

else

printf("\t");

}

}

int leastused(){

int i,min=0,n=0;

for(i=1;i<fsize;i++)

if(arrive[i]<arrive[min]){

min=i;

n++;

}

if(n==0)

return 0;

else

return min;

}

int pagelocation(int pageno){

int i,flag=0;

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

if(frame[i]==pageno){

flag=1;

break;

}

return i;

}

**Output:**

