```
// Program  program to illustrate process system calls
#include<stdio.h>
#include<unistd.h>
int main()
{
int pid;
pid=fork();
if (pid==0)
{
printf(" this is child with id %d\n",getpid());
printf(" the Parent id is % d\n ", getppid());
sleep(6);
printf(" again this is child -  with id %d\n", getpid());
printf(" the parent id is %d\n", getppid());
}
else
{
printf("this is parent with id %d\n", getpid());
wait();
printf("this is parent after child exits with id %d\n", getpid());
}
}
```

Output

```
this is parent with id 2782
 this is child with id 2783
 the Parent id is  2782
  again this is child -  with id 2783
 the parent id is 2782
this is parent after child exits with id 2782
```

```
// Program 2 program to illustrate I/O system Calls
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<fcntl.h>

int main (void)
{
   int fd[2];
   char buf1[12] = "hello world";
   char buf2[12];

   // assume sample.txt is already created
   fd[0] = open("sample.txt", O_RDWR);
   fd[1] = open("sample.txt", O_RDWR);

   write(fd[0], buf1, strlen(buf1));
   write(1, buf2, read(fd[1], buf2, 12));

   close(fd[0]);
   close(fd[1]);

   return 0;

}
```

Output

```
hello world
```

```
/*PROGRAM 3A: PROGRAM TO SEND HELLO */
#include<sys/msg.h>
#include<sys/ipc.h>
#include<sys/types.h>
#include<stdio.h>
#include<string.h>
#include<unistd.h>
main()
{
 int p1,p2,c;
 char mtext[10];
 p1=msgget((key_t)110,IPC_CREAT|0666);
 p2=msgget((key_t)100,IPC_CREAT|0666);
 strncpy(mtext,"hello",5);
 if(msgsnd(p2,&mtext,5,0)==-1);
 perror("MESSAGE SENT WAS FAILED");
 printf("\nP2 is going to receive the message\n");
 if(msgrcv(p1,&mtext,9,0,0)==-1)
 perror("message was not received");
 printf("message received %s \n",mtext);
}
```

```
/*PROGRAM 3B:PROGRAM TO SEND THANK YOU*/
#include<sys/msg.h>
#include<sys/ipc.h>
#include<sys/types.h>
#include<stdio.h>
#include<string.h>
#include<unistd.h>
main()
{
 int  i,p1,p2,c;
 char mtext[10];
 p1=msgget((key_t)110,IPC_CREAT|0666);
 p2=msgget((key_t)100,IPC_CREAT|0666);
 if(msgrcv(p2,&mtext,5,0,0)==-1)
 perror("failure");
 printf("message received %s\n",mtext);
 strncpy(mtext,"thank you",9);
 if(msgsnd(p1,&mtext,9,0)==-1)
 perror("fail");
}
```

Output:]$ cc program3a.c
$cc program3b.c
$./a.out
message received hello
$cc program3a.c
$./a.out
P2 is going to receive the message
message received thank you

```
/* PROGRAM 4 FIRST COME FIRST SERVED ALGORITHM*/
#include<stdio.h>
struct process
{
float bt;
float tt;
float wt;
};
main()
{
int n,i;
float twt=0,ttt=0,awt,att;
struct process p[15];
printf("\n\n\t\t FIRST COME FIRST SERVED ALGORITHM");
printf("\n enter the no.of process");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
printf("enter burst time for %d",i);
scanf("%f",&p[i].bt);
}
p[1].wt=0;
twt=twt+p[1].wt;
p[1].tt=p[1].bt;
ttt=p[1].tt;
for(i=2;i<=n;i++)
{
p[i].wt=p[i-1].tt;
twt=twt+p[i].wt;
p[i].tt=p[i].wt+p[i].bt;
ttt=ttt+p[i].tt;
}
printf("\n\n\n\t\t the total wait time is...:%f",twt);
printf("\n\n\n\t\t the total turn around time is...:%f",ttt);
awt=twt/n;
att=ttt/n;
printf("\n\n\t\t the average wait time is...:%f",awt);
printf("\n\n\n\n\t\t the average turn around time is...:%f",att);
}
```

FIRST COME FIRST SERVED ALGORITHM

enter the no.of process 3
enter burst time for 1 24
enter burst time for 2 3
enter burst time for 3 3

the total wait time is...:51.000000

the total turn around time is...:81.000000

the average wait time is...:17.000000

the average turn around time is...:27.000000

```
/* Program 5.SHORTEST JOB FIRST ALGORITHM */
#include<stdio.h>
struct process
{
float bt;
float tt;
float wt;
};
main()
{
int n,i,j;
float twt=0,ttt=0,awt,att,temp;
struct process p[15];
printf("\n\n\n\t shortest job first algorithm");
printf("\n enter the no of processes");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
printf("enter burst time for %d",i);
scanf("%f",&p[i].bt);
}
for(i=1;i<=n-1;i++)
{
for(j=i+1;j<=n;j++)
{
if(p[i].bt>p[j].bt)
{
temp=p[i].bt;
p[i].bt=p[j].bt;
p[j].bt=temp;
}
}
}
p[1].wt=0;
twt=twt+p[1].wt;
p[1].tt=p[1].bt;
ttt=p[1].tt;
for(i=2;i<=n;i++)
{
p[i].wt=p[i-1].tt;
twt=twt+p[i].wt;
p[i].tt=p[i].wt+p[i].bt;
ttt=ttt+p[i].tt;
```

```
}
printf("\n\n\n\t the total wait time is....%f",twt);
printf("\n\n\n\n\t the total around time is....%f",ttt);
awt=twt/n;
att=ttt/n;
printf("\n\n\n\t the average wait time is...%f",awt);
printf("\n\n\n\t the average turn around time is...%f",att);
}
```

shorted job first algorithm

```
enter the no of processes3
enter burst time for 1 24
enter burst time for 2  3
enter burst time for 3  3


        the total wait time is....9.000000



        the total around time is....39.000000


        the average wait time is...3.000000


        the average turn around time is...13.000000
```

```
/* PROGRAM 6 PRIORITY CPU SCHEDULING ALGORITHM */
#include<stdio.h>
struct process
{
float bt;
float tt;
float wt;
int pri;
};
main()
{
int n,x,i,j;
float twt=0,ttt=0,awt,att,temp,temp1;
struct process p[15];
printf("\n\n\n\t Priority job scheduling algorithm");
printf("\n enter the no of processes");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
printf("enter burst time for %d\n",i);
scanf("%f",&p[i].bt);
printf("enter priority for %d\n",i);
scanf("%d",&p[i].pri);
}

for(i=1;i<=n-1;i++)
{
for(j=i+1;j<=n;j++)
{
if(p[i].pri > p[j].pri)
{
temp=p[i].bt;
temp1=p[i].pri;
p[i].bt=p[j].bt;
p[i].pri=p[j].pri;
p[j].bt=temp;
p[j].pri=temp1;
}
}
}
p[1].wt=0;
twt=twt+p[1].wt;
```

```
p[1].tt=p[1].bt;
ttt=p[1].tt;
for(i=2;i<=n;i++)
{
p[i].wt=p[i-1].tt;
twt=twt+p[i].wt;
p[i].tt=p[i].wt+p[i].bt;
ttt=ttt+p[i].tt;
}
printf("\n\n\n\t the total wait time is....%f",twt);
printf("\n\n\n\n\t the total around time is....%f",ttt);
awt=twt/n;
att=ttt/n;
printf("\n\n\n\t the average wait time is...%f",awt);
printf("\n\n\n\t the average turn around time is...%f",att);
}
```

Priority job scheduling algorithm

```
 enter the no of processes3
enter burst time for 1
24
enter priority for 1
3
enter burst time for 2
3
enter priority for 2
1
enter burst time for 3
4
enter priority for 3
2


    the total wait time is....10.000000


    the total around time is....41.000000


    the average wait time is...3.333333


    the average turn around time is...13.666667[root@home ~]#
```

```
/* Program 7 ROUND ROBIN ALGORITHM*/
#include<stdio.h>
struct process
{
float bt;
float rbt;
float wt;
float tt;
float lst;
};
main()
{
struct process p[10];
float awt=0,att=0;
int i,n;
float ttt=0,twt=0,cp=0;
int pp=0;
int count,flag,tq;
printf("\n\n\n\t ROUND ROBIN ALGORITHM");
printf("\n\n\t ENTER THE NUMBER OF PROCESSES.....");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
printf("\n enter burst time for process%d\t\t\t",i);
scanf("%f",&p[i].bt);
p[i].rbt=p[i].bt;
p[i].lst=p[i].wt=0;
}
printf("\n\n enter the time quantum.....");
scanf("%d",&tq);
do
{
flag=0;
for(i=1;i<=n;i++)
if((count=p[i].rbt)>0)
{
flag=1;
count=((count>=tq)?tq:count);
cp=cp+count;
p[i].rbt-=count;
if(pp!=i)
```

```
{
pp=i;
p[i].wt+=cp-p[i].lst-count;
p[i].lst=cp;
}
}
}
while(flag);
for(i=1;i<=n;i++)
{
p[i].tt=p[i].lst;
twt+=p[i].wt;
ttt+=p[i].wt+p[i].bt;
}
awt=twt/n;
att=ttt/n;
printf("\n\n\n\t total waiting time=%f", twt);
printf("\n\n\n\t total turn around time=%f",ttt);
printf("\n\n\n\t average waiting time is=%f",awt);
printf("\n\n\n\t average turn around time is=%f",att);
}
```

ROUND ROBIN ALGORITHM

ENTER THE NUMBER OF PROCESSES.....3

enter burst time for process1 24
enter burst time for process2 3
enter burst time for process3 3

enter the time quantum.....4

total waiting time=17.000000

total turn around time=47.000000

average waiting time is=5.666667

average turn around time is=15.666667

```
/*Program 8 IPC USING PIPES*/
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
int pfds[2];
char buf[30];
pipe(pfds);
if(!fork())
{
printf("\n\t\t\t IPC using PIPES \n\n");
printf("CHILD: writing to the pipe\n");
write(pfds[1],"PERIYAR ARTS COLLEGE",20);
printf("child:exiting\n");
exit(0);
}
else
{
printf("PARENT:reading from pipe\n");
read(pfds[0],buf,20);
printf("PARENT: reading from pipe as %s\n",buf);
wait(NULL);
}
}
```

IPC using PIPES

```
CHILD: writing to the pipe
child:exiting
PARENT:reading from pipe
PARENT: reading from pipe as PERIYAR ARTS COLLEGEª
```

```
/* Program 9 First Fit Algorithm */
#include<stdio.h>
#include<process.h>
void main()
{
        int a[20],p[20],i,j,n,m;
        printf("Enter no of Blocks.\n");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("Enter the %dst Block size:",i);
                scanf("%d",&a[i]);
        }
        printf("Enter no of Process.\n");
        scanf("%d",&m);
        for(i=0;i<m;i++)
        {
                printf("Enter the size of %dst Process:",i);
                scanf("%d",&p[i]);
        }
      for(i=0;i<n;i++)
        {
        for(j=0;j<m;j++)
                {
                        if(p[j]<=a[i])
                        {
                                printf("The Process %d allocated to %d\n",j,a[i]);
                                p[j]=10000;
                                break;
                        }
                }
}
        for(j=0;j<m;j++)
        {
        if(p[j]!=10000)
                {
        printf("The Process %d is not allocated\n",j);
                }
        }
```

}

OUTPUT:

Enter no of Blocks.
5
Enter the 0st Block size:500
Enter the 1st Block size:400
Enter the 2st Block size:300
Enter the 3st Block size:200
Enter the 4st Block size:100
Enter no of Process.
5
Enter the size of 0st Process:100
Enter the size of 1st Process:350
Enter the size of 2st Process:400
Enter the size of 3st Process:150
Enter the size of 4st Process:200
The Process 0 allocated to 500
The Process 1 allocated to 400
The Process 3 allocated to 200
The Process 2 is not allocated
The Process 4 is not allocated

shell script for factorial of a number

```
#factorial using while loop

echo "Enter a number"
read num

fact=1

while [ $num -gt 1 ]
do
  fact=$((fact * num))
  num=$((num - 1))
done

echo $fact
```

Output

```
Enter a number
3
6
Enter a number
4
24
Enter a number
5
120
```

```
// Program 12  To generate Fibanocci series
echo "How many number of terms to be generated ?"
   read n
   x=0
   y=1
   i=2
   echo "Fibonacci Series up to $n terms :"
   echo "$x"
   echo "$y"
   while [ $i -lt $n ]
   do
   i=`expr $i + 1 `
   z=`expr $x + $y `
   echo "$z"
   x=$y
   y=$z
   done

   OUTPUT:-
   How many number of terms to be generated ? 5
   Fibonacci Series up to $n terms :
   0
   1
   1
   2
   3
```