***Program 1:***

#include<stdio.h>

#include<stdlib.h>

int main()

{

System("clear");

System("echo a.The CPU type and Model and Model Name:");

System("cat /proc/cpuinfo | awk 'NR==3 {print $4}' " );

system("cat /proc/cpuinfo | awk 'NR==4 {print $3}' ");

system("cat /proc/cpuinfo | awk 'NR==5 {print $4 $5 $6 $7 $8 $9}' ");

system("echo OS Release ");

system(" cat /proc/sys/kernel/osrelease");

system("echo UPTIME");

system("cat /proc/uptime");

system("echo Amt of time spent in user mode");

system(" cat /proc/stat | awk 'NR==1 {print $2}'");

system("echo Amt of time spent in kernel mode");

system(" cat /proc/stat | awk 'NR==1 {print $4}'");

system("echo No of disk request");

system(" cat /proc/stat | awk 'NR==10 {print $1 $2}'");

system("echo No of context siwtches");

system(" cat /proc/stat | awk 'NR==11 {print $2}'");

system("echo time at which last booted");

system(" cat /proc/stat | awk 'NR==12 {print $2}'");

system("echo Memory Total");

system(" cat /proc/meminfo | awk 'NR==1 {print $2 $3}'");

system("echo Memory Free");

system(" cat /proc/meminfo | awk 'NR==2 {print $2 $3}'");

system("echo Load Avg");

system(" cat /proc/loadavg | awk 'NR==1 {print $3 $5}'");

return 0;

}

**PROGRAM 2:**

#include<time.h>

#include<fcntl.h>

#include<stdlib.h>

#include<stdio.h>

#include<sys/stat.h>

main(argc,argv)

int argc;

char \*argv[];

{

struct stat statbuff;

int fd;

fd=open(argv[1],O\_RDONLY|O\_CREAT,0744);

fstat(fd,&statbuff);

close(fd);

printf("user id is %d",statbuff.st\_uid);

printf("group id is %d",statbuff.st\_gid);

printf(" size is %d",statbuff.st\_size);

printf("no. of links %d",statbuff.st\_nlink);

printf("arrival time %d",statbuff.st\_atime);

printf(" arrival time in correct format %d",ctime(&statbuff.st\_atime));

return 0;

}

***Program 3:***

#include<stdio.h>

#include<fcntl.h>

#include<stdlib.h>

char buffer[256];

main(argc,argv)

int argc;

char \*argv[];

{

int fdold,fdnew;

if(argc!=3)

{

printf("Need 2 arguments for copy ");

exit(1);

}

fdold=open(argv[1],O\_RDONLY);

if(fdold==-1)

{

printf("\n The file can not be opened");

exit(1);

}

fdnew=creat(argv[2],0666);

if(fdnew==-1)

{

printf("File can not be created ");

exit(1);

}

copy(fdold,fdnew);

exit(0);

}

copy(old,new)

int old,new;

{

int count;

while((count=read(old,buffer,sizeof(buffer)))>0)

write(new,buffer,count);

}

**Program 4a.c**

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

int pid;

pid=fork();

if(pid==-1)

{

printf("\n forking failed ");

exit(-1);

}

else

{

printf("\n Executing Code pid %d",pid);

exit(0);

}

}

**Program 4b.c**

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

int pid;

pid=fork();

if(pid==-1)

{

printf("\n forking failed ");

exit(-1);

}

else if(pid==0)

{

printf("\n I am child process ");

printf("\n My pid is : %d ",getpid());

printf("\n The value of pid :%d",pid);

exit(0);

}

else

{

printf("\n I am parent ");

printf("\n My pid is :%d",getpid());

printf("\n The value of pid : %d", pid);

exit(0);

}

}

**Program 4c.c**

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

int pid;

pid=fork();

if(pid==-1)

{

printf("\n forking failed ");

exit(-1);

}

else if(pid==0)

{

printf("\n I am child process ");

execlp("cal","cal",NULL);

exit(0);

}

else

{

printf("\n I am parent ");

printf("\n My pid is :%d",getpid());

printf("\n The value of pid : %d", pid);

exit(0);

}

}

**Program 4c\_1.c**

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

int pid;

pid=fork();

if(pid==-1)

{

printf("\n forking failed ");

exit(-1);

}

else if(pid==0)

{

printf("\n I am child process ");

execlp("./hello.bin","hello.bin",NULL);

exit(0);

}

else

{

printf("\n I am parent ");

printf("\n My pid is :%d",getpid());

printf("\n The value of pid : %d", pid);

exit(0);

}

}

**//hello world for bin program**

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

printf("\n hello World ");

exit(0);

}

**Program 4d.c**

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

int pid;

pid=fork();

if(pid==-1)

{

printf("\n forking failed ");

exit(-1);

}

else if(pid==0)

{

printf("\n I am child process ");

printf("\n My pid is %d",getpid());

exit(0);

}

else if(pid>0)

{

printf("\n I am parent ");

printf("\n My pid is :%d",getpid());

wait(NULL);

printf("\n I was waiting");

exit(0);

}

}

**Program 5:**

#include <iostream>

using namespace std;

#define buffersize 10

typedef struct

{

int pid;

int bt;

}item;

item buffer[buffersize];

int in=0;

int out=0;

void producer()

{

item np;

cout<<"\n Enter process id:";

cin>>np.pid;

cout<<"\n Enter burst :";

cin>>np.bt;

if(((in+1)%buffersize)==out)

{

cout<<"\n Fully scheduled";

}

else

{

buffer[in]=np;

in=(in+1)%buffersize;

}

}

void consumer()

{

item nc;

if(in==out)

{

cout<<"\n No item produced";

}

else

{

nc=buffer[out];

out=(out+1)%buffersize;

cout<<"\n Consumed item :";

cout<<"PID:"<<nc.pid;

cout<<"\n BURST:"<<nc.bt;

}

}

int main()

{

int ch;

char c='y';

do

{

cout<<"\n MENU :";

cout<<"\n 1. PRODUCE ITEM(S)";

cout<<"\n 2. CONSUME ITEM(S)";

cout<<"\n Enter your choice:";

cin>>ch;

switch(ch)

{

case 1: while(c=='y')

{

producer();

cout<<"\n MORE?";

cin>>c;

}

Break;

case 2: while(c=='y')

{

consumer();

cout<<"\n MORE ?";

cin>>c;

}

break;

}

cout<<"\n Do you want to continue ?";

cin>>c;

}while(c=='y'||c=='Y');

return 0;

}

**Program 6:**

#include<stdio.h>

#include<sys/types.h>

#include<string.h>

#include<unistd.h>

#define BUFFER\_SIZE 25

#define READ\_END 0

#define WRITE\_END 1

int main()

{

char write\_msg[BUFFER\_SIZE]="Greetings";

char read\_msg[BUFFER\_SIZE];

int fd[2];

pid\_t pid;

if(pipe(fd)==-1)

{

printf("Pipe Failed");

return 1;

}

pid=fork();

if(pid<0)

{

printf("Fork Failed");

return 1;

}

if(pid>0)

{

close(fd[READ\_END]);

write(fd[WRITE\_END],write\_msg,strlen(write\_msg)+1);

close(fd[WRITE\_END]);

}

else

{

close(fd[WRITE\_END]);

read(fd[READ\_END],read\_msg,BUFFER\_SIZE);

printf("read %s",read\_msg);

close(fd[READ\_END]);

}

return 0;

}

**Program 7:**

//threads

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

#include<pthread.h>

int sum;

void \*runner( void \*param);

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

{

pthread\_t tid;

pthread\_attr\_t attr;

if(argc!=2)

{

printf("Usage \n");

return -1;

}

if(atoi(argv[1]) < 0)

{

printf("It must be greater than 0");

return -1;

}

pthread\_attr\_init(&attr);

pthread\_create(&tid,&attr,runner,argv[1]);

pthread\_join(tid,NULL);

printf("sum=%d",sum);

return 0;

}

void \*runner(void \*param)

{

int i, upper=atoi(param);

sum=0;

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

sum+=i;

pthread\_exit(0);

}

**Program 7\_fact:**

//threads

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

#include<pthread.h>

int fact;

void \*runner( void \*param);

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

{

pthread\_t tid;

pthread\_attr\_t attr;

if(argc!=2)

{

printf("Usage \n");

return -1;

}

if(atoi(argv[1]) < 0)

{

printf("It must be greater than 0");

return -1;

}

pthread\_attr\_init(&attr);

pthread\_create(&tid,&attr,runner,argv[1]);

pthread\_join(tid,NULL);

printf("FACTORIAL=%d \n",fact);

return 0;

}

void \*runner(void \*param)

{

int i, upper=atoi(param);

fact=1;

//printf("The value is :%d", upper);

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

fact=fact\*i;

pthread\_exit(0);

}

**PROGRAM 7\_FIB.C**

//threads

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

#include<pthread.h>

int fib;

void \*runner( void \*param);

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

{

pthread\_t tid;

pthread\_attr\_t attr;

if(argc!=2)

{

printf("Usage \n");

return -1;

}

if(atoi(argv[1]) < 0)

{

printf("It must be greater than 0");

return -1;

}

pthread\_attr\_init(&attr);

pthread\_create(&tid,&attr,runner,argv[1]);

pthread\_join(tid,NULL);

return 0;

}

void \*runner(void \*param)

{

int i,next, upper=atoi(param);

fib=0;

i=1;

printf("\n FIBONACCI SERIES ...", upper);

while(fib<=upper)

{

printf("%d ",fib);

next=i;

i=fib+i;

fib=next;

}

pthread\_exit(0);

}

**Program 7\_2threads:**

//2 THREADs

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

#include<pthread.h>

int osum,esum,sum;

void \*oddsum( void \*param);

void \*evensum(void \*param);

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

{

pthread\_t tid;

pthread\_attr\_t attr;

if(argc!=2)

{

printf("Usage \n");

return -1;

}

if(atoi(argv[1]) < 0)

{

printf("It must be greater than 0");

return -1;

}

pthread\_attr\_init(&attr);

pthread\_create(&tid,&attr,oddsum,argv[1]);

pthread\_create(&tid,&attr,evensum,argv[1]);

pthread\_join(tid,NULL);

sum=esum+osum;

printf("Odd value sum :%d",osum);

printf("Even value sum: %d",esum);

printf("Total sum=%d",sum);

return 0;

}

void \*oddsum(void \*param)

{

int i, upper=atoi(param);

osum=0;

for(i=1;i<=upper;i=i+2)

osum+=i;

pthread\_exit(0);

}

void \*evensum(void \*param)

{

int i, upper=atoi(param);

esum=0;

for(i=0;i<=upper;i=i+2);

esum+=i;

pthread\_exit(0);

}

**Program 8A:**

#include<iostream>

using namespace std;

class node

{

public:

int bt;//burst time

int et;//execution time

node\* next;

node(int b=0)

{

bt=b;

et=0;

next=NULL;

}

};

class link\_list

{

node\* head;

node\* tail;

int pno;

public:

link\_list()

{

head=tail=NULL;

}

void insertion();

void fcfs();

void traverse();

};

void link\_list::insertion()

{

int i,b;

cout<<"\n Enter the number of processes:";

cin>>pno;

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

{

cout<<"\n Enter burst time:";

cin>>b;

node\* n=new node(b);

if(head==NULL)

{

head=tail=n;

}

else

{

tail->next=n;

tail=n;

}

}

}

void link\_list::fcfs()

{

int p=0;

node \*n=head;

while(n!=NULL)

{

n->et=p+n->bt;

p=n->et;

n=n->next;

}

}

void link\_list::traverse()

{

int tt,wt,i;

node\* n;

n=head;

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

{

cout<<i;

cout<<" "<<n->bt;

tt=n->et;

cout<<" "<<tt;

wt=tt-n->bt;

cout<<" "<<wt;

cout<<endl;

n=n->next;

}

}

int main()

{

link\_list L;

L.insertion();

L.fcfs();

L.traverse();

return 0;

}

**Program 8a\_varat:**

#include<iostream>

using namespace std;

class node

{

public:

int at;

int bt;//burst time

int et;//execution time

node\* next;

node(int b=0,int a=0)

{

bt=b;

et=0;

next=NULL;

at=a;

}

};

class link\_list

{

node\* head;

node\* tail;

int pno;

public:

link\_list()

{

head=tail=NULL;

}

void insertion();

void fcfs();

void traverse();

};

void link\_list::insertion()

{

int i,b,a;

cout<<"\n Enter the number of processes:";

cin>>pno;

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

{

cout<<"\n Enter burst time:";

cin>>b;

cout<<"\n Enter arrival time:";

cin>>a;

node\* n=new node(b,a);

if(head==NULL)

{

head=tail=n;

}

else

{

tail->next=n;

tail=n;

}

}

}

void link\_list::fcfs()

{

int p=0;

node \*n=head;

while(n!=NULL)

{

n->et=p+n->bt;

p=n->et;

n=n->next;

}

}

void link\_list::traverse()

{

int tt,wt,i;

node\* n;

n=head;

cout<<"\n PNO"<<" AT "<<" BT "<<" TT "<<" WT ";

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

{

cout<<i;

cout<<" "<<n->at;

cout<<" "<<n->bt;

tt=n->et-n->at;

cout<<" "<<tt;

wt=tt-n->bt;

cout<<" "<<wt;

cout<<endl;

n=n->next;

}

}

int main()

{

link\_list L;

L.insertion();

L.fcfs();

L.traverse();

return 0;

}

**Program 8b :**

#include<iostream>

using namespace std;

class node

{

public:

int at;

int bt;//burst time

int et;//execution time

node\* next;

node(int b=0,int a=0)

{

bt=b;

et=0;

next=NULL;

at=a;

}

};

class link\_list

{

node\* head;

node\* tail;

int pno;

public:

link\_list()

{

head=tail=NULL;

}

void insertion();

void fcfs();

void traverse();

};

void link\_list::insertion()

{

int i,b,a;

cout<<"\n Enter the number of processes:";

cin>>pno;

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

{

cout<<"\n Enter burst time:";

cin>>b;

cout<<"\n Enter arrival time:";

cin>>a;

node\* n=new node(b,a);

if(head==NULL)

{

head=tail=n;

}

else if(n->bt < head->bt)

{

n->next=head;

head=n;

}

else if(tail->bt < n->bt)

{

tail->next=n;

tail=n;

}

else

{

node \*temp, \*prev;

temp=head->next;

prev=head;

if(temp->next!=NULL && temp->bt<n->bt)

{

prev=temp;

temp=head->next;

}

prev->next=n;

n->next=temp;

}

}

}

void link\_list::fcfs()

{

int p=0;

node \*n=head;

while(n!=NULL)

{

n->et=p+n->bt;

p=n->et;

n=n->next;

}

}

void link\_list::traverse()

{

int tt,wt,i;

node\* n;

n=head;

cout<<"\n PNO"<<" AT "<<" BT "<<" TT "<<" WT \n";

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

{

cout<<i;

cout<<" "<<n->at;

cout<<" "<<n->bt;

tt=n->et-n->at;

cout<<" "<<tt;

wt=tt-n->bt;

cout<<" "<<wt;

cout<<endl;

n=n->next;

}

}

int main()

{

link\_list L;

L.insertion();

L.fcfs();

L.traverse();

return 0;

}

**Program 8c:**

#include <iostream>

using namespace std;

class srtf

{

int at[100],bt[100],pt[100],bt2[100],ex[100];

int et[100];

int ptt,pno;

public:

void insertion();

void srt();

void display();

};

void srtf::insertion()

{

ptt=0;

cout<<"\n Enter number of processes:";

cin>>pno;

for(int i=0;i<pno;i++)

{

cout<<"\n Enter arrival time:";

cin>>at[i];

pt[i]=at[i];

cout<<"\n Enter burst time";

cin>>bt[i];

bt2[i]=bt[i];

ptt+=bt[i];

}

bt[pno]=ptt;

}

void srtf::srt()

{

int i,j,smallest;

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

{

smallest=pno;//smallest point to the index of bt array which contains to

tal burst time and no process will have burst time largere than it

for(j=0;j<pno;j++)

{

if(pt[j]<=i)

{

if(bt[smallest]>bt[j] && bt[j]!=-1)

{

smallest=j;

}

}

}

et[i]=smallest;

pt[smallest]++;

bt[smallest]--;

if(bt[smallest]==0)

{

ex[smallest]=i+1;

bt[smallest]=-1;

}

}

}

void srtf::display()

{

int tt,wt;

cout<<"\n GANTT CHART ";

for(int j=0;j<ptt;j++)

{

cout<<et[j];

}

cout<<"\n pno"<<" TT"<<" WT";

for(int i=0;i<pno;i++)

{

tt=ex[i]-at[i];

wt=tt-bt2[i];

cout<<"\n"<<i<<" "<<ex[i]<<" "<<tt<<" "<<wt;

}

}

int main()

{

srtf S;

S.insertion();

S.srt();

S.display();

return 0;

}

**Program 8d:**

#include<iostream>

using namespace std;

class node

{

public:

int bt;//burst time

int et;//execution time

int pr;

node\* next;

node(int p,int b=0)

{

pr=p;

bt=b;

et=0;

next=NULL;

}

};

class link\_list

{

node\* head;

node\* tail;

int pno;

public:

link\_list()

{

head=tail=NULL;

}

void insertion();

void fcfs();

void traverse();

};

void link\_list::insertion()

{

int i,b,p;

cout<<"\n Enter the number of processes:";

cin>>pno;

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

{

cout<<"\n Enter burst time:";

cin>>b;

cout<<"\n Enter priority";

cin>>p;

node\* n=new node(p,b);

if(head==NULL)

{

head=tail=n;

}

else if(head->pr>p)

{

n->next=head;

head=n;

}

else if(p>tail->pr)

{

tail->next=n;

tail=n;

}

else

{

node \*temp,\*prev;

temp=head->next;

prev=head;

if(temp->next!=NULL && temp->pr<n->pr)

{

prev=temp;

temp=temp->next;

}

prev->next=n;

n->next=temp;

}

}

}

void link\_list::fcfs()

{

int p=0;

node \*n=head;

while(n!=NULL)

{

n->et=p+n->bt;

p=n->et;

n=n->next;

}

}

void link\_list::traverse()

{

int tt,wt,i;

node\* n;

n=head;

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

{

cout<<i;

cout<<" "<<n->bt;

tt=n->et;

cout<<" "<<tt;

wt=tt-n->bt;

cout<<" "<<wt;

cout<<endl;

n=n->next;

}

}

int main()

{

link\_list L;

L.insertion();

L.fcfs();

L.traverse();

return 0;

}

**Program 8e.c:**

#include <iostream>

using namespace std;

class srtf

{

int at[100],bt[100],pt[100],bt2[100],ex[100],pr[100];

int et[100];

int ptt,pno;

public:

void insertion();

void srt();

void display();

};

void srtf::insertion()

{

ptt=0;

cout<<"\n Enter number of processes:";

cin>>pno;

for(int i=0;i<pno;i++)

{

cout<<"\n Enter arrival time:";

cin>>at[i];

pt[i]=at[i];

cout<<"\n Enter burst time";

cin>>bt[i];

bt2[i]=bt[i];

cout<<"\n Enter priority:";

cin>>pr[i];

ptt+=bt[i];

}

pr[pno]=9999;

}

void srtf::srt()

{

int i,j,smallest;

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

{

smallest=pno;//smallest point to the index of bt array which contains to

tal burst time and no process will have burst time largere than it

for(j=0;j<pno;j++)

{

if(pt[j]<=i)

{

if(pr[smallest]>pr[j] && bt[j]!=-1)

{

smallest=j;

}

}

}

et[i]=smallest;

pt[smallest]++;

bt[smallest]--;

if(bt[smallest]==0)

{

ex[smallest]=i+1;

bt[smallest]=-1;

}

}

}

void srtf::display()

{

cout<<"\n pno"<<" TT"<<" WT";

int tt,wt;

for(int j=0;j<ptt;j++)

{

cout<<et[j];

}

for(int i=0;i<pno;i++)

{

tt=ex[i]-at[i];

wt=tt-bt2[i];

cout<<"\n"<<i<<" "<<ex[i]<<" "<<tt<<" "<<wt;

}

}

int main()

{

srtf S;

S.insertion();

S.srt();

S.display();

return 0;

}

**Program 8f.cc:**

#include <iostream>

using namespace std;

class rr

{

int bt[100],at[100];

int ts,ptt,pno;

public:

void insertion();

void rref();

};

void rr::insertion()

{

int i=0;

ptt=0;

cout<<"\n Enter the number of processes";

cin>>pno;

cout<<"\n Enter time slice:";

cin>>ts;

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

{

cout<<i+1;

cout<<"\n Enter Arrival Time:";

cin>>at[i];

cout<<"\n Enter burst :";

cin>>bt[i];

ptt+=bt[i];

}

}

void rr::rref()

{

int set=0;

int i=0;

while(set<ptt)

{

if(at[i]>set)

i=(i+1)%pno;

else

{

if(bt[i]!=0)

{

cout<<"\n";

cout<<(i+1);

if(bt[i]>ts)

{

bt[i]=bt[i]-ts;

set+=ts;

}

else

{

set+=bt[i];

bt[i]=0;

}

cout<<" "<<bt[i]<<" "<<set;

}

i=(i+1)%pno;

}

}

}

int main()

{

rr R;

R.insertion();

R.rref();

return 0;

}

**Program 10:**

#include<iostream>

using namespace std;

int page\_size, pmsize, phy\_mem[100], frame\_table[100],total\_frame;

int k=0;

class pt

{

public:

int pid, fno[10], p\_size, nframes;

void input\_function();

void pt\_function();

void page\_function();

int retpid();

void deallocation();

};

void pt::page\_function()

{

int p\_no;

cout<<"Enter page no; ";

cin>> p\_no;

cout<<fno[p\_no];

}

void pt::deallocation()

{

for(int i=0; i< nframes; i++)

{

frame\_table[fno[i]]=0;

}

cout<<"\n Frames Deallocated";

}

void pt::input\_function()

{

cout<<"\n Enter PID";

cin>> pid;

cout<<"\n Enter process size";

cin>>p\_size;

nframes= p\_size/page\_size;

if(p\_size%page\_size!=0)

nframes++;

cout<<"\n Number of frames : " << nframes ;

}

void pt::pt\_function()

{

int j=0,flag=0;

for(; k<total\_frame && j < nframes; k++)

{

if( frame\_table[k] ==0)

{

flag=1;

fno[j]=k;

j++;

frame\_table[k]==1;

}

}

if(flag==0)

{

cout<<"\n No empty frames in memory !!!!!";

}

}

int pt::retpid()

{

return pid;

}

int main()

{

pt P[10];

int np,i,ch,pid;

char c;

cout<<"\n Enter page size:";

cin>> page\_size;

cout<<"\n enter Physical memory size:";

cin>> pmsize;

total\_frame=pmsize/page\_size;

cout<<"\n Total frames"<< total\_frame;

cout<<"\n How many processes you want to create :";

cin>>np;

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

{

P[i].input\_function();

P[i].pt\_function();

}

do

{

cout<<"\n MENU";

cout<<"\n 1. ENTER A PAGE NUMBER AND RETRIEVE ITS FRAME INFORMATION";

cout<<"\n 2. DEALLOCATE FRAMES ";

cout<<"\n Enter your choice:";

cin>>ch;

cout<<"\n Enter a PID";

cin>>pid;

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

{

if(pid==P[i].retpid())

break;

}

if(i>=np)

cout<<"\n NO process with this pid exist !!!";

else

{

switch(ch)

{

case 1: P[i].page\_function();

break;

case 2:P[i].deallocation();

break;

}

}

cout<<"\n Do you want to continue ?";

cin>>c;

}while(c=='y'||c=='Y');

return 1;

}

**Program 11:**

#include<time.h>

#include<fcntl.h>

#include<stdlib.h>

#include<stdio.h>

#include<sys/stat.h>

//#define s\_IFREG 0100000

//#define s\_IFDIR 0040000

//#define s\_IFCHR 0020000

//#define s\_IFBLK 0060

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

{

struct stat statbuf;

int fd;

fd=open(argv[1],O\_RDONLY|O\_CREAT,0744);

fstat(fd,&statbuf);

close(fd);

if(S\_ISREG(statbuf.st\_mode))

printf("Regular file");

if(S\_ISDIR(statbuf.st\_mode))

printf("Directory");

if(S\_ISCHR(statbuf.st\_mode))

printf("Character file");

if(S\_ISBLK(statbuf.st\_mode))

printf("Block fILE");

return 0;

}

Extras:

[u4110@linux6 ~]$ cat fork1.c

#include<stdio.h>

#include<stdlib.h>

int main()

{

int pid;

pid=fork();

pid=fork();

printf("MY :%d, Parent :%d",getpid(),getppid());

printf("\n ");

return 0;

}