|  |
| --- |
|  |
| Operating System |
| Laboratory |
| Manish Kumar Gupta  260/CO/08  COE-I |

|  |
| --- |
|  |

**Contents**

1. CPU Scheduling Algorithm

1. Shortest Job First
2. First Come First Serve
3. Priority Scheduling
4. Round Robin

2. Memory Management Algorithm

1. Best Fit
2. First Fit
3. Worst Fit

3. Page Replacement Algorithm

1. First In First Out
2. Least Recently Used
3. Optimal

4. Disk Scheduling Algorithm

1. SSTF
2. FCFS
3. C-SCAN

5. System Calls Fork(),, Exit()

**CPU Scheduling Algorithm**

#include<stdio.h>

#include<conio.h>

int name[10],time[10],prior[10];

#define swap(a,b) a=b-a+(b=a)

int main()

{

int a,b,i,x,y,q,done;

int wait[10],res[10],turn[10];

printf("\nProgram for Job Scheduling Algorithms(all Jobs arrive at same time):-");

printf("\nEnter no. of jobs(max:10) ");

scanf("%d",&y);

for(i=0;i<10;i++)

{ wait[i]=res[i]=turn[i]=0;

name[i]=time[i]=prior[i]=0;

}

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

{ printf("\nEnter job name, burst time ");

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

}

printf("\nEnter your choice of algorithm ");

printf("\n1.Shortest Job First ");

printf("\n2.First Come First Serve ");

printf("\n3.Priority Scheduling ");

printf("\n4.Round Robin ");

scanf("%d",&x);

switch(x)

{ case 1:

for(a=0;a<y-1;a++)

for(b=a+1;b<y;b++)

{ if(time[a]>time[b])

{ swap(name[a],name[b]);

swap(time[a],time[b]);

}

}

turn[0]=time[0];

for(a=1;a<y;a++)

{ wait[a]=0;

for(b=0;b<a;b++)

wait[a]+=time[b];

res[a]=wait[a];

turn[a]=wait[a]+time[a];

}

printf("\n\tName\tWaiting\tResponse\tTurn Around time");

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

{ printf("\n\t%3d\t%4d\t%4d\t\t%8d",name[i],wait[i],res[i],turn[i]);}

break;

case 2:

turn[0]=time[0];

for(a=1;a<y;a++)

{ wait[a]=0;

for(b=0;b<a;b++)

wait[a]+=time[b];

res[a]=wait[a];

turn[a]=wait[a]+time[a];

}

printf("\n\tName\tWaiting\tResponse\tTurn Around time");

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

{ printf("\n\t%3d\t%4d\t%4d\t\t%8d",name[i],wait[i],res[i],turn[i]);}

break;

case 3:

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

{ printf("\nEnter job priority: ");

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

}

for(a=0;a<y-1;a++)

for(b=a+1;b<y;b++)

{ if(prior[a]>prior[b])

{ swap(name[a],name[b]);

swap(time[a],time[b]);

swap(prior[a],prior[b]);

}

}

turn[0]=time[0];

for(a=1;a<y;a++)

{ wait[a]=0;

for(b=0;b<a;b++)

wait[a]+=time[b];

res[a]=wait[a];

turn[a]=wait[a]+time[a];

}

printf("\n\tName\tWaiting\tResponse\tTurn Around time");

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

{ printf("\n\t%3d\t%4d\t%4d\t\t%8d",name[i],wait[i],res[i],turn[i]);}

break;

case 4:

printf("\nEnter quantum size: ");

scanf("%d",&q);

turn[0]=time[0];

done=time[0];

for(a=1;a<y;a++)

{ res[a]=a\*q-(time[a-1]>q?0:-time[a-1]+q);

turn[a]=time[a];

done+=time[a];

}

for(i=0,b=0;b<done;i++)

{ a=i%y;

if(time[a]>0)

{ b+=(time[a]>q?q:time[a]);

time[a]-=q;

wait[a]=b-turn[a];

}

}

for(a=0;a<y;a++)

{ turn[a]+=wait[a];

}

printf("\n\tName\tWaiting\tResponse\tTurn Around time");

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

{ printf("\n\t%4d\t%4d\t%4d\t\t%8d",name[i],wait[i],res[i],turn[i]);}

break;

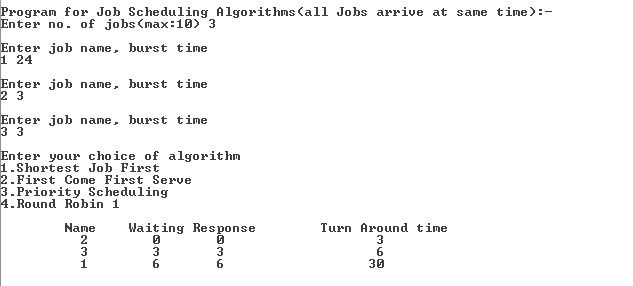
default:printf("\nWrong choice");

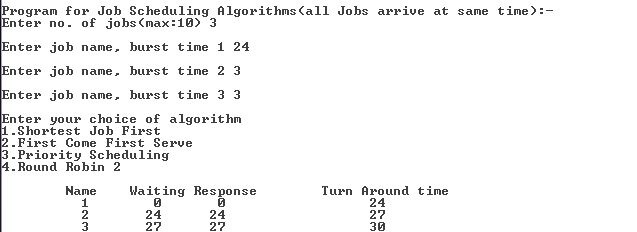
}

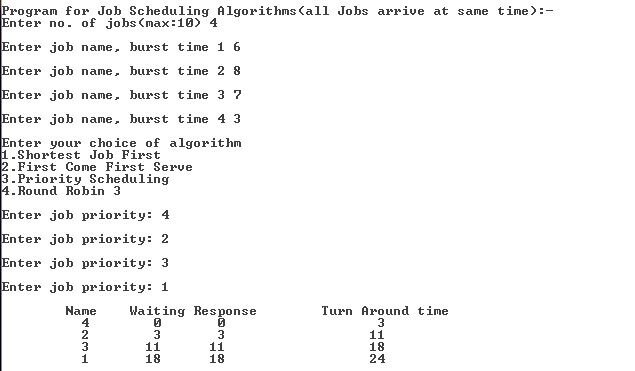
getch();

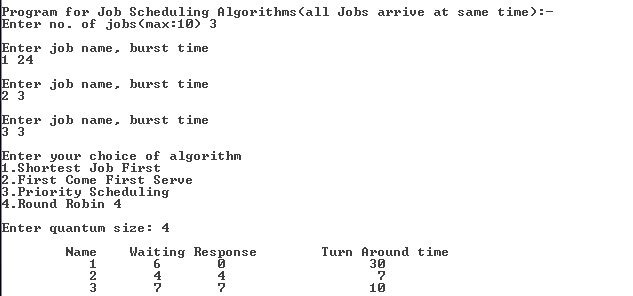
return 0;

}









**Memory Management Algorithm**

#include<stdio.h>

#include<conio.h>

void sort(int \*mem,int n)

{ int i, j, temp;

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

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

{

if (\*(mem+j) < \*(mem+j+1))

{

temp = \*(mem+j);

\*(mem+j) = \*(mem+j+1);

\*(mem+j+1) = temp;

}

}

}

void xsort(int \*mem,int n)

{ int i, j, temp;

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

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

{

if (\*(mem+j) > \*(mem+j+1))

{

temp = \*(mem+j);

\*(mem+j) = \*(mem+j+1);

\*(mem+j+1) = temp;

}

}

}

int main()

{

int ch, i=0, j=0, n, x, mem[100],allocate[50],flag=0 ;

printf("Enter the no of available memory blocks");

scanf("%d",&n);

for(i=0;(i<n) && (i<100);i++)

{ printf("Enter size of %d block",i+1);

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

}

printf("The mem blocks are:\n");

for(i=0;(i<n) && (i<100);i++)

printf("%d ->",mem[i]);

printf("Enter the no of mem blocks to be allocated");

scanf("%d",&x);

for(i=0;(i<x) && (i<50);i++)

{ printf("Enter size of %d block to be allocated",i+1);

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

}

printf("\nEnter your choice of algorithm ");

printf("\n1.Best Fit ");

printf("\n2.First Fit ");

printf("\n3.Worst Fit ");

scanf("%d",&ch);

switch(ch)

{ case 1:xsort(mem,n);

for(i=0;(i<x) && (i<50);i++)

{ j=0;

flag = 0;

while((j<n) && (j<100))

{ if (allocate[i] <= mem[j])

{ mem[j] = mem[j] - allocate[i];

xsort(mem,n);

flag = 1;

break;}

else j++;

}

if (!flag) printf("unable to allocate block of size %d\n",allocate[i]);

else printf("allocated mem block no %d of size %d\n", j, allocate[i]);

}

break;

case 2:

for(i=0;(i<x) && (i<50);i++)

{ j=0;

flag = 0;

while((j<n) && (j<100))

{ if (allocate[i] <= mem[j])

{ mem[j] = mem[j] - allocate[i];

flag = 1;

break;}

else j++;

}

if (!flag) printf("unable to allocate block of size %d\n",allocate[i]);

else printf("allocated mem block no %d of size %d\n", j, allocate[i]);

}

break;

case 3:sort(mem,n);

for(i=0;(i<x) && (i<50);i++)

{ j=0;

flag = 0;

while((j<n) && (j<100))

{ if (allocate[i] <= mem[j])

{ mem[j] = mem[j] - allocate[i];

sort(mem,n);

flag = 1;

break;}

else j++;

}

if (!flag) printf("unable to allocate block of size %d\n",allocate[i]);

else printf("allocated mem block no %d of size %d\n", j, allocate[i]);

}

break;

default:printf("\nWrong choice");

}

printf("The mem blocks are:\n");

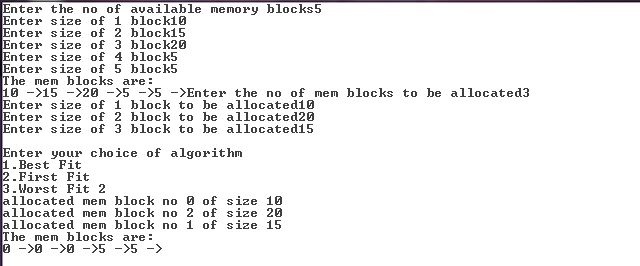
for(i=0;(i<n) && (i<100);i++)

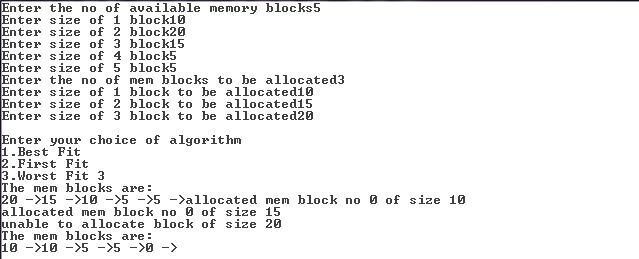
printf("%d ->",mem[i]);

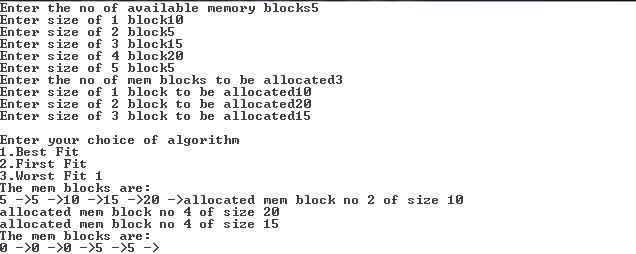
getch();

return 0;

}







**Page Replacement Algorithm**

**FIFO**

#include <stdio.h>

#include<conio.h>

#define FRAMES 3

int queue[FRAMES];

int fr\_search(int page\_no,int n)

{

int flag =0,j ;

int check\_frames = (FRAMES < n) ? FRAMES : n ;

for( j=0; j< check\_frames ;j++ )

if(page\_no==queue[j])

{ flag=1;

break;

}

return flag ;

}

int main()

{

int n,i,miss=0,j,flag=0,k=0;

int page[10];

printf("Enter no of Page: max 10 page IDs");

scanf("%d",&n);

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

{

printf("Enter %d page", (i+1));

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

}

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

{

flag = fr\_search(page[i],n);

if (flag == 0 )

{

queue[k]=page[i];

miss++;

k = (k+1)%FRAMES;

}

}

if(n<=FRAMES)

{ printf("\ntotal no. of page fault is %d,no need for page replacement\n",miss);

return 1;

}

else

{

for( ;i<n;i++ )

{ flag = fr\_search(page[i],FRAMES) ;

if (flag==0)

{

queue[k]=page[i];

k=(k+1)%FRAMES;

miss ++ ;

}

}

}

printf("\nTotal no. of page faults are %d \n",miss);

getch();

return 0;

}



**LRU**

#include <stdio.h>

#include<conio.h>

int frame[3]={-1,-1,-1};

int count[3]={0,0,0};

int miss=0;

int replace(int frame[],int count[])

{int max=count[0],i=1,j=0;

while(i<3)

{if(count[i]>max)

{max = count[i];

j=i;

}

i++;

}

return j;

}

int search(int frame[],int page, int k)

{ int j = -1,i;

for(i=0;i<3;i++)

if(frame[i] == page)

return i;

if(k != 3)

{miss++;

return k;

}

return j;

}

int main()

{ int i,j,k,n,index;

int page[10];

printf("Enter page string length:max 10");

scanf("%d",&n);

for(i=0;(i<n) && (i<10);i++)

{printf("Enter %d page",i+1);

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

}

for(i=0;(i<n) && (i<10);i++)

{k=0;

while(k<3)

{if(frame[k] < 0)

break;

else{count[k]++; k++;}

}

index=search(frame,page[i],k);

if((index < 0) && (k == 3))

{j=replace(frame,count);

miss++;

count[j]=0;

frame[j]=page[i];

}

else

{count[index]=0;

frame[index]=page[i];

}

}

printf("No of faults%d",miss);

getch();

return 0;

}



**Optimal**

#include <stdio.h>

#include<conio.h>

int frame[3]={-1,-1,-1};

int age[3]={32760,32760,32760};

int miss=0;

int search(int arr[], int p, int k)

{ int i, j=-1;

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

if(arr[i] == p)

return i;

return j;

}

void replace(int frame[],int page[],int i,int n)

{ int j=0,k,curr,max;

while(j<3)

{ curr=frame[j];

for(k=i;(k<n) && (k<10);k++)

if(page[k] == curr)

{age[j]=k;

break;}

j++;

}

k=0;

max = age[0];

for(j=1; j<3; j++)

if(max < age[j])

{k = j;

max=age[j];}

frame[k] = page[i];

}

int main()

{ int i, page[10], n, k=0,index;

printf("Enter page string:max 10");

scanf("%d",&n);

for(i=0; (i<n) && (i<10); i++)

{printf("Enter %d page",i+1);

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

}

for(i=0;(i<n) && (i<10) && (k<3);i++)

{index = search(frame,page[i],k);

if((index < 0) || (k == 0))

{frame[k++]=page[i];

miss++;}

}

for(; (i<n) && (i<10); i++)

{index = search(frame,page[i],k);

if(index<0)

{replace(frame,page,i,n);

miss++;

}

}

printf("Total page faults: %d",miss);

getch();

return 0;

}



**Disk Scheduling Algorithm**

**SSTF**

#include <stdio.h>

#include <conio.h>

int main()

{ int head, disk[10], diff[10], i, n, min=32760, j=0, k=0;

int acc[10]={0,0,0,0,0,0,0,0,0,0};

printf("enter the no of disk access:max 10");

scanf("%d",&n);

for(i=0;(i<n) && (i<10);i++)

{printf("Enter the %d access",i+1);

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

}

printf("Enter the current head position");

scanf("%d",&head);

while((k<n) &&(k<10))

{min=32760;

for(i=0;(i<n) && (i<10);i++)

{if(diff[i] != 32760)

diff[i] = head-disk[i];

if(diff[i]<0)

diff[i] = (-1)\*diff[i];

}

for(i=0;(i<n) && (i<10);i++)

if(diff[i] < min)

{min = diff[i];

j=i;

}

diff[j]=32760;

acc[k++]=disk[j];

head=disk[j];

}

printf("The access sequence is\n");

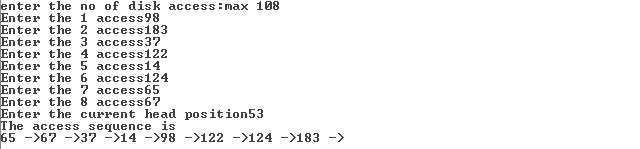
for(k=0;(k<n) && (k<10);k++)

printf("%d ->",acc[k]);

getch();

return 0;

}



**FCFS**

#include<stdio.h>

#include<conio.h>

int main()

{ int i,n,disk[10];

printf("Enter the no of disk accesses:max 10");

scanf("%d",&n);

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

{ printf("Enter the %d access",i+1);

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

}

printf("The Squence of disk access is ");

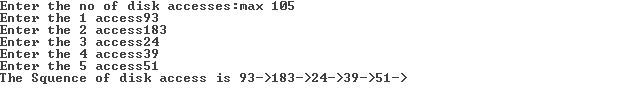
for(i=0;i<n;i++)

printf("%d->",disk[i]);

getch();

return 0;

}



**CSCAN**

#include<stdio.h>

#include<conio.h>

int main()

{ int head,i,disk[10],n,temp,j,acc[10],k;

printf("Entr the current head position");

scanf("%d",&head);

printf("Enter the disk access:max 10");

scanf("%d",&n);

for(i=0;(i<n) && (i<10);i++)

{printf("Enter the %d access",i+1);

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

}

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

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

{ if (disk[j] > disk[j+1])

{temp = disk[j];

disk[j] = disk[j+1];

disk[j+1] = temp;

}

}

for(i=0;(i<n) && (i<10);i++)

if(head < disk[i])

{ j=i;

break;}

for(i=j,k=0;(i<n) && (i<10);i++,k++)

acc[k]=disk[i];

for(i=0;(i<j);i++,k++)

acc[k]=disk[i];

printf("The access sequence is");

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

printf("%d ->",acc[k]);

getch();

return 0;

}



**System Calls Fork(), Exit()**

#include <stdlib.h>

#include<stdio.h>

#include<conio.h>

int globalVariable = 2;

int main()

{

char\* sIdentifier;

pid\_t pID = fork();

if (pID == 0) // child

{ sIdentifier = "Child Process: ";

globalVariable++;

}

else if (pID < 0) // failed to fork

{

printf("Failed to fork\n");

exit(1);

}

else // parent

{sIdentifier = "Parent Process:";

}

printf("%s",sIdentifier);

printf("Global variable:%d \n",globalVariable);

getch();

return 0;

}

**OUTPUT:**

Parent Process: Global variable: 2

Child Process: Global variable: 3