**MEMORY ALLOCATION METHODSFIRST FIT, WORST FIT, BEST FIT**

SANJANA S NAIR

CSE C

33

#include<stdio.h>

#define MAX 100

int temp,q[MAX],allocated[MAX];

void display(int m[],int n,int pr[],int c,int p)

{

int j,totalfrag=0,frag;

for(j=0;j<n+c;j++)

{

printf(" ");

}

printf("\n");

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

{

temp=(j<c)?pr[q[j]-1]:m[j];

printf("|%3d ",temp);

frag=m[j]-temp;

totalfrag+=frag;

if(frag!=0)

printf("+%3d ",frag);

}

printf("|\n");

for(j=0;j<n+c;j++)

{

printf(" ");

}

printf("\n");

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

{

printf(" P%d\t ",q[j]);

allocated[q[j]-1]=1;

}

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

{

if(allocated[i]!=1)

printf("\nProcess %d not allocated\n",i+1);

}

printf("\n");

printf("Total internal fragmentation = %d\n",totalfrag);

}

void firstfit(int m[],int n,int pr[],int p)

{

int i,j,c=0;

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

{

allocated[i]=0;

for(j=c;j<n;j++)

{

if(m[j]>=pr[i])

{

temp=m[j];

m[j]=m[c];

m[c]=temp;

q[c]=i+1;

c++;

Break;

}

}

}

display(m,n,pr,c,p);

}

void bestfit(int m[],int n,int pr[],int p)

{

int j,c=0,i;

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

allocated[i]=0;

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

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

temp=m[j];

m[j]=m[j+1];

m[j+1]=temp;

}

}

}

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

for(j=c;j<n;j++){

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

temp=m[j];

m[j]=m[c];

m[c]=temp;

q[c]=i+1;

c++;}

}

}

display(m,n,pr,c,p);

}

void worstfit(int m[],int n,int pr[],int p)

{

int j,c=0,i;

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

{

allocated[i]=0;

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

{

if(m[j]<m[j+1])

{

temp=m[j];

m[j]=m[j+1];

m[j+1]=temp;

}

}

}

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

for(j=c;j<n;j++){

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

temp=m[j];

m[j]=m[c];

m[c]=temp;

q[c]=i+1;

c++;

}

}

}

display(m,n,pr,c,p);

}

void main()

{

int i,n,ch,pr[MAX],p,m[MAX];

printf("Enter the number of blocks:");

scanf("%d",&n);

printf("Enter the mem blocks:\n");

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

{

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

}

printf("Enter the number of processes:");

scanf("%d",&p);

printf("Enter the process mem:\n");

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

{

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

}

do

{

printf("\nEnter 1:Firstfit 2:Bestfit 3:Worstfit 4:Exit :\n");

scanf("%d",&ch);

switch(ch)

{

case 1 : firstfit(m,n,pr,p);

break;

case 2: bestfit(m,n,pr,p);

break;

case 3: worstfit(m,n,pr,p);

break;

case 4: break;

default:printf("Invalid input\n");

}

}while(ch!=4);

}

**SAMPLE OUTPUT**

Enter the number of blocks:4

Enter the mem blocks:

100

200

300

400

Enter the number of processes:3

Enter the process mem:

50

120

350

Enter 1:Firstfit 2:Bestfit 3:Worstfit 4:Exit :

1

| 50 + 50 |120 + 80 |350 + 50 |300 |

P1 P2 P3

Total internal fragmentation = 180

Enter 1:Firstfit 2:Bestfit 3:Worstfit 4:Exit :

2

| 50 + 50 |120 + 80 |350 + 50 |300 |

P1 P2 P3

Total internal fragmentation = 180

Enter 1:Firstfit 2:Bestfit 3:Worstfit 4:Exit :

3

| 50 +350 |120 +180 |200 |100 |

P1 P2

Process 3 not allocated

Total internal fragmentation = 530

Enter 1:Firstfit 2:Bestfit 3:Worstfit 4:Exit :

4