OPERATING SYSTEM LAB **Nithin Jose R4 46**

Assignment No: 7

Memory allocation methods

CODE:-

```
#include<stdio.h>
void firstfit(int *block,int *process,int nb,int np){
int i,j,flag[10],allocation[10],t=0,pflag[10];
for(i=0;i<10;i++){
flag[i]=0;
pflag[i]=0;
allocation[i]=-1;
}
for(i=0;i<np;i++)
for(j=0;j<nb;j++)
if(flag[j]==0 && block[j]>=process[i])
{
allocation[j]=i;
pflag[i]=1;
flag[j]=1;
break;
}
for(i=0;i<nb;i++)
printf("___________________");
printf("\n");
for(i=0;i<nb;i++){
printf("|\t%d\t|",block[i]);
}
printf("\n");
for(i=0;i<nb;i++)
printf("___________________");
printf("\n");
for(i=0;i<nb;i++){
if(allocation[i]>=0)
printf("|\t%d\t|",process[allocation[i]]);
else printf("|\tNA\t|");
}
printf("\n");
```

```
for(i=0;i<nb;i++)
printf("____________________");
printf("\n");
for(i=0;i<np;i++)
if(pflag[i]==0){
t++;
printf("%d ",process[i]);
}
if(t==1)
printf("sized process is unallocated");
if(t>1)
printf("sized processess are unallocated");
}
void worstfit(int *block,int *process,int nb,int np){
int i,j,max,maxj,flag[10],allocation[10],t=0,pflag[10];
for(i=0;i<10;i++){
flag[i]=0;
pflag[i]=0;
allocation[i]=-1;
}
for(i=0;i<np;i++){
max=0;
for(j=0;j<nb;j++)
if(flag[j]==0&& block[j]>=process[i]&&block[j]>max)
{ max=block[j];
maxj=j;
}
if(max>0){
allocation[maxj]=i;
pflag[i]=1;
flag[maxj]=1;
}
}
for(i=0;i<nb;i++)
printf("____________________");
printf("\n");
for(i=0;i<nb;i++){
printf("|\t%d\t|",block[i]);
}
printf("\n");
for(i=0;i<nb;i++)
printf("____________________");
printf("\n");
```

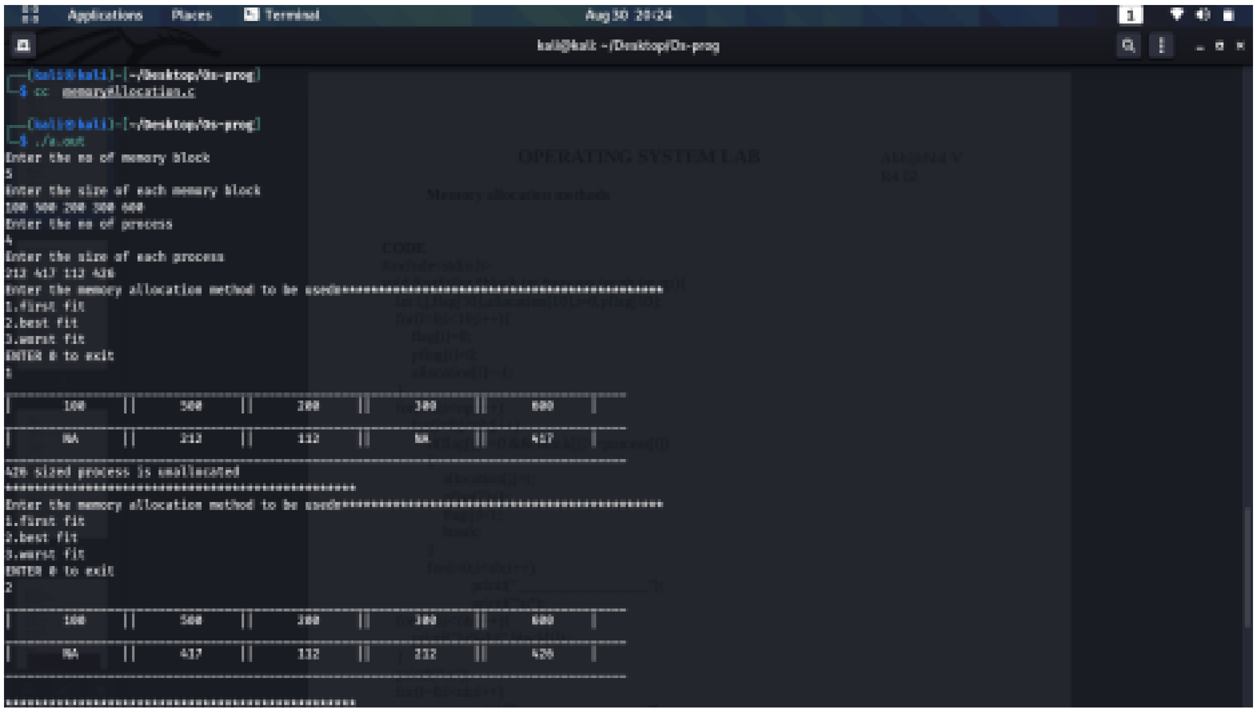
```
for(i=0;i<nb;i++){
if(allocation[i]>=0)
printf("|\t%d\t|",process[allocation[i]]);
else printf("|\tNA\t|");
}
printf("\n");
for(i=0;i<nb;i++)
printf("___________________");
printf("\n");
for(i=0;i<np;i++)
if(pflag[i]==0){
t++;
printf("%d ",process[i]);
}
if(t==1)
printf("sized process is unallocated");
if(t>1)
printf("sized processess are unallocated");
}
void bestfit(int *block,int *process,int nb,int np){
int i,j,max,maxj,flag[10],allocation[10],t=0,pflag[10];
for(i=0;i<10;i++){flag[i]=0;
pflag[i]=0;
allocation[i]=-1;
}
for(i=0;i<np;i++){
max=100000000;
for(j=0;j<nb;j++){
if(flag[j]==0&& block[j]>=process[i]&&block[j]<max)
{ max=block[j];
maxj=j;
}
}
if(max<100000000){
allocation[maxj]=i;
pflag[i]=1;
flag[maxj]=1;
}
}
for(i=0;i<nb;i++)
printf("___________________");
printf("\n");
for(i=0;i<nb;i++){
```

```
printf("|\t%d\t|",block[i]);
}
printf("\n");
for(i=0;i<nb;i++)
printf("__________________");
printf("\n");
for(i=0;i<nb;i++){
if(allocation[i]>=0)
printf("|\t%d\t|",process[allocation[i]]);
else printf("|\tNA\t|");
}
printf("\n");
for(i=0;i<nb;i++)
printf("__________________");
printf("\n");
for(i=0;i<np;i++)
if(pflag[i]==0){
t++;
printf("%d ",process[i]);
}
if(t==1)
printf("sized process is unallocated");
if(t>1)
printf("sized processess are unallocated");
}

void nextfit(int *block,int *process,int nb,int np)
{
int i,j=0,flag[10],allocation[10],t=0,pflag[10];
for(i=0;i<10;i++){
flag[i]=0;
pflag[i]=0;
allocation[i]=-1;
}
for(i=0;i<np;i++)
while(j<nb)
 {
   if(flag[j]==0 && block[j]>=process[i])
  {

   allocation[j]=i;
   pflag[i]=1;
   flag[j]=1;
```

```
    break;

   }

   j = (j + 1) % nb;

 }
 for(i=0;i<nb;i++)
 printf("___________________");
 printf("\n");
 for(i=0;i<nb;i++){
 printf("|\t%d\t|",block[i]);
 }
 printf("\n");
 for(i=0;i<nb;i++)
 printf("___________________");
 printf("\n");
 for(i=0;i<nb;i++){
 if(allocation[i]>=0)
 printf("|\t%d\t|",process[allocation[i]]);
 else printf("|\tNA\t|");
 }
 printf("\n");
 for(i=0;i<nb;i++)
 printf("___________________");
 printf("\n");
 for(i=0;i<np;i++)
 if(pflag[i]==0){
 t++;
 printf("%d ",process[i]);
 }
 if(t==1)
 printf("sized process is unallocated");
 if(t>1)
 printf("sized processess are unallocated");
 }
 void main(){
 int i,block[10],process[10],nb,np,select;
 printf("Enter the no of memory block\n");
 scanf("%d",&nb);
 printf("Enter the size of each memory block\n");for(i=0;i<nb;i++){
 scanf("%d",&block[i]);
 }
```

```
printf("Enter the no of process\n");
scanf("%d",&np);
printf("Enter the size of each process\n");
for(i=0;i<np;i++){
scanf("%d",&process[i]);
}
while(1){
printf("Enter the memory allocation method to be used\
n*******************************************\n");
printf("1.first fit\n2.best fit\n3.worst fit\nENTER 0 to exit\n");
scanf("%d",&select);
if(select==1){
firstfit(block,process,nb,np);
}
else if(select==2){
bestfit(block,process,nb,np);
}
else if(select==3){
worstfit(block,process,nb,np);
}
else if(select==4){
nextfit(block,process,nb,np);
}
else if(select==0){
break;
}
printf("\n***********************************************\n");
}
}
```

Output



```
┌──(kali㉿kali)-[~/Desktop/Os-prog]
└─$ cc  memoryAllocation.c

┌──(kali㉿kali)-[~/Desktop/Os-prog]
└─$ ./a.out
Enter the no of memory block
5
Enter the size of each memory block
100 500 200 300 600
Enter the no of process
4
Enter the size of each process
212 417 112 426
Enter the memory allocation method to be used**********************************************
1.first fit
2.best fit
3.worst fit
ENTER 0 to exit
1
------------------------------------------------------------------------------------
|      100     ||      500     ||      200     ||      300     ||      600     |
------------------------------------------------------------------------------------
|      NA      ||      212     ||      112     ||      NA      ||      417     |
------------------------------------------------------------------------------------
426 sized process is unallocated
************************************************
Enter the memory allocation method to be used**********************************************
1.first fit
2.best fit
3.worst fit
ENTER 0 to exit
2
------------------------------------------------------------------------------------
|      100     ||      500     ||      200     ||      300     ||      600     |
------------------------------------------------------------------------------------
|      NA      ||      417     ||      112     ||      212     ||      426     |
------------------------------------------------------------------------------------

************************************************
```